DISTRIBUTED MULTI-AGENT BASED TRAFFIC MANAGEMENT SYSTEM

Balaji Parasumanna Gokulan

B.E., University of Madras

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE
2011

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr.Dipti Srinivasan without whose guidance, support, and encouragement it would have been impossible for me to finish this work. I would like to thank Dr.Lee Der-Horng and Dr. P.Chandrashekar for their help and guidance during my research work.

I would also like to thank all my colleagues in the lab for making it an ideal environment to perform research. My special thanks goes to Mr.Seow Hung Cheng, who took extra effort to ensure all the facilities, equipments and software are available to us at all time.

My stay in Singapore would not have been fun-filled without my friends. Some of my friends who deserve a special mention are: Vishal Sharma, Krishna Agarwal, Krishna Mainali, R.P.Singh, Sahoo Sanjib Kumar, D.Shyamsundar, Raju Gupta, J.Sundaramurthy, Anupam Trivedi and Atul Karande. The fun filled discussions ranging from politics to movies at Technoedge canteen every evening, the intense tennis sessions and joint music lessons we had together will stay as a sweet memory for my entire lifetime.

I would like to thank my wife Soumini for her patience and support during the final thesis writing phase. My acknowledgement would be incomplete without a special mention of my parents and sister. I am greatly indebted to my parents and my sister for their support and unconditional love they showered during my entire PhD studies.

Last but not least, I gratefully acknowledge the financial support offered by National University of Singapore during the course of my postgraduate studies in Singapore.

TABLE OF CONTENTS

ABSTRACT

Traffic congestion is a major recurring problem faced in many countries in the world due to increased urbanization and availability of affordable vehicles. Congestion problem can be dealt with in a number of ways – Increasing the capacity of the roads, promoting alternate modes of transportation or making efficient use of the existing infrastructure. Among these, the most feasible option is to improve the usage of existing roads. Adjustment of the green time in signals to allow more vehicles to cross the intersection has been the widely accepted method for solving congestion problem. Green time essentially dictates the time during which vehicles are allowed to cross an intersection, thereby avoiding conflicting movements of vehicles and improving safety at an intersection.

Conventional and traditional traffic signal control methods have shown limited success in optimizing the timings in signals due of the lack of accurate mathematical models of traffic flow at an intersection and uncertainties associated with the traffic data. Traffic flow refers to the number of vehicles crossing an intersection every hour. The traffic environment is dynamic and traffic signal timings at one intersection influences the traffic flow rate at the connected intersection. This necessitates the use of hybrid computational intelligent models to predict the traffic flow and influence of the neighbouring intersection signals on the green signal timings. Increased communication overheads, reliability issues, data mining, and real-time control requirements limits the use of centralized traffic signal controls. These limitations are overcome by distributed traffic signal controls. However, a major disadvantage with distributed signal control is the partial view of each computing entity involved in the calculation of green time at an intersection. In order to improve the global view, communication and learning capabilities needs to be incorporated in the computing entity to create a model of the neighbouring computing entities. Multi-agent systems provide such an distributed architecture with learning and communication capabilities.

In this dissertation, a distributed multi-agent architecture capable of learning from the traffic environment and communicating with the neighbouring intersections is developed. Four computational intelligent decision systems with different internal architectures were developed. First two approaches were offline trained methods using deductive reasoning. The third approach was based on online batch learning method to co-evolve the membership functions and rule base in type-2 fuzzy decision system. The fourth decision system developed is an online shared reward Q-learning based neuro-type2 fuzzy network.

Performance of the proposed multi-agent based traffic signal controls for different traffic simulation scenarios were evaluated using a simulated urban road traffic network of Singapore. Comparative analysis performed over the benchmark traffic signal controls – Hierarchical Multi-agent Systems (HMS) and GLIDE (Green Link Determine) indicated considerable improvement in travel time delay and mean speed of vehicles when using proposed multi-agent based traffic signal control methods.

LIST OF FIGURES

LIST OF TABLES

LIST OF DEFINITIONS

Green time Duration or period of time during which vehicles in a lane are allowed to cross an intersection.

Phase A signal phase can be defined as an unique set of traffic signal movements, where a movement is controlled by a number of traffic signal lights that changes colour at one time.

Cycle The time required for one full cycle of signal indications, given in seconds.

Cycle length Time taken to complete all phases at an intersection. Cycle time includes the green time, amber time and all red time of every phase in use at an intersection.

Right of way Lanes with green signals to allow the flow of vehicles.

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

Offset Time lag between the start of green time in a phase of signals at nearby connected intersections to allow free flow of vehicles without facing any red signal.

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.

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

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
MAS	Multi-Agent System
HMS	Hierarchical Multi-agent System
GLIDE	Green Link Determining system
T2DR	Type-2 Fuzzy Deductive Reasoning decision system
GFMAS	Geometric Fuzzy Multi-Agent System
QLT2	Q-Learning neuro-Type2 fuzzy decision system
QLT1	Q-Learning neuro-Type1 fuzzy decision system
SET2	Symbiotic Evolutionary Type-2 fuzzy decision system
GAT2	Genetic algorithm tuned Type-2 fuzzy decision system
SCATS	Sydney Coordinated Adaptive Traffic System
SCOOT	Split Cycle Offset Optimization Technique
FIPA	Foundation for Intelligent Physical Agents
ACL	Agent Communication Language
RL	Reinforcement Learning

CHAPTER 1

INTRODUCTION

Traffic congestion is a major recurring problem faced in many countries due to the increased level of urbanization and the availability of cheaper vehicles. One of the options to reduce congestion is to construct newer infrastructure to accommodate the increased vehicle count. However, it is highly infeasible in developing countries where space is a major constraint. Second most feasible option is improving the usage of the existing roads through optimization of traffic signal timings. This can alleviate the congestion levels experienced at intersections by evenly distributing the travel delay among all the vehicles, thereby reducing the travel time of vehicles inside the road network and providing a temporal separation for vehicles with *right of way* in a link.

Traffic signal controls the movement of traffic by adjusting the *split* of each phase assigned in a total *cycle time* and by modifying the *offset*. *Split* refers to the total time allocated to each phase in a cycle, *right of way* refers to the lanes with green signal and allowable movement during a specific phase, and *offset* is the time lag between the start of green time for successive intersections, which is required to ensure a free flow of vehicles (progression) with minimum wait time along a specific direction. The breakdown of a three-phase cycle at an intersection is shown in Figure 1.1 to elucidate the terms split, phase, cycle length, offset, progression and right of way.

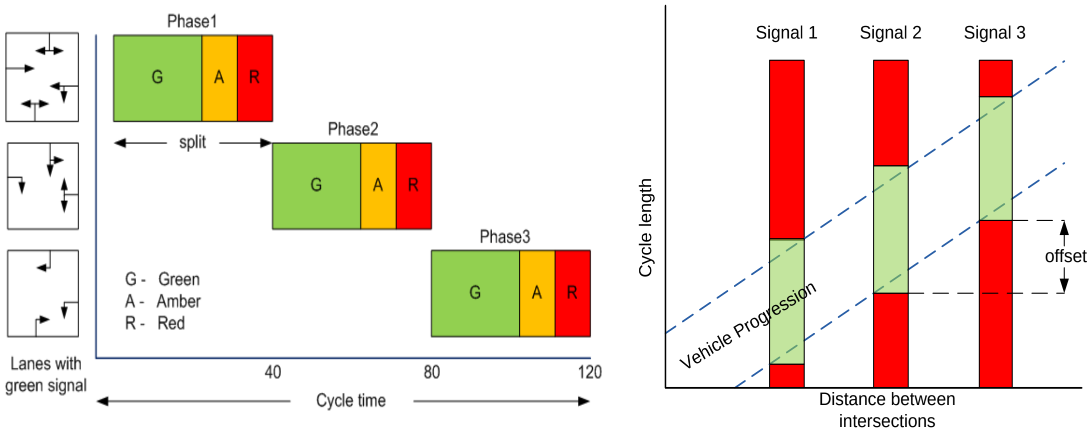


Figure 1.1. Typical three phase traffic signal cycle time indicating phase splits and right of way

Traffic signal timing optimization or split adjustment to change the green time of a phase maximizes the throughput of the vehicles at the controlled intersection and helps in maintaining the degree of saturation of all the links connected to the intersection without compromising the safety of vehicles inside the road network. Computing an optimal value of green time in a phase is an extremely complex task as the signal timings at the intersection affects the traffic flow in the connected intersections.

Early traffic signal control schemes were typically designed for isolated intersections, as these form the basic components of road traffic network and can be easily modelled. Based on the type of control used, the traffic signal controls can be classified into three types:

- Pre-timed or Fixed control
- Traffic responsive Control
- Traffic adaptive control

One of the first mathematical models developed for calculating the green time with an objective to reduce the average delay experienced by vehicles inside a road network was proposed in [1] and formed the basis for the fixed time traffic signal controls. The green time of each phase in a signal was calculated offline using historical traffic flow pattern collected from the urban arterial roads. The designed traffic controller was not capable of handling any sudden variations in the traffic from the pattern used to calculate the green time. Further, offline estimation methods are prone to losses when switching between signal plans, especially with rapid traffic changes.

In order to overcome these limitations, traffic responsive methods that changes the signal timings based on the traffic experienced at the intersection were introduced. Though these signal controls improved the traffic congestion over fixed time signal controls, lack of ability to foresee the traffic condition, faulty sensors, and environmental conditions affect its performance.

Traffic adaptive methods are intelligent traffic signal control methods with an ability to predict the traffic flow and adjust the timings online. Based on the type of architecture used, the traffic adaptive methods can be classified into two types.

- Centralized control
- Semi-distributed control
- Distributed Control

Centralized traffic signal controls determine the network wide signal timings at a central location. The traffic data collected from each intersection is sent to a central server that compute the timings required at each intersection for the specific traffic flow experienced at the intersection. Centralized traffic controls require large amount of traffic data to be communicated from the intersection to the control centre. This increases the communication overhead to a large extent. Further, raw data sent from the intersection needs to be sorted and ordered according to the phase timing calculation thereby increasing the computational overhead. The performance is also affected because of the traffic data loss and addition of noise to the data.

Semi-distributed traffic signal controls improved the reliability of the traffic signal controls by using hierarchical structure. Though the communication cost is lesser than in centralized control, the cost is still substantially high. With increase in the traffic network size, the control becomes complex and difficult to handle.

In distributed traffic signal controls, traffic signal at each intersection needs to be controlled by a computing entity. The signal timings for the intersection are computed autonomously using the local data collected from the sensors connected to the intersection. However, the restricted view of the sensors limits the traffic view available to each computing entity. In order to improve the global traffic view and improve the performance of the signal control, the controls need to learn, communicate, and adapt dynamically. This requirement is satisfied by the multi-agent systems with hybrid computational intelligent decision systems with communication capabilities. Computational intelligent methods are required as only approximate mathematical models of traffic flow at an arterial intersection are available.

1.1. BRIEF OVERVIEW OF MULTI-AGENT SYSTEMS

An agent can be viewed as a self-contained, concurrently executing thread of control that encapsulates some state, and communicates with its environment, and possibly other agents through some sort of message passing [2] between agents. Agent-based systems offer advantages where independently developed components must interoperate in a heterogeneous environment, e.g., the internet. Agent-based systems are increasingly applied in a wide range of areas including telecommunications, BPM (Business process modelling), computer games, distributed system control and robotic systems. The significant advantage of the agent system in contrast to simple distributed problem solving is that the environment is an integral part of the agent.

Multi-Agent Systems(MAS) is a branch of distributed artificial intelligence that emphasizes the joint behaviour of agents with some degree of autonomy and complexities arising from their interactions. Multi-agent systems allow the sub-problems of a constraint satisfaction problem to be subcontracted to different problem solving agents with their own individual interests and goals. This increases the speed of operation, creates parallelism, and reduces the risk of system collapse due to single point failure. Though generalized multi-agent platform could be used for solving different problems, it is a common practise to design a tailor made multi-agent architecture according to the application. Multi-agent systems are able to synergistically combine the various computational intelligent techniques for attaining a superior performance by combining the advantages of various techniques into a single framework. MAS also provides extra degree of freedom to model the behaviour of the system to be as competitive or coordinating, with each method having its own merits and demerits.

1.2. MAIN OBJECTIVES OF THE RESEARCH

The main objective of this dissertation is to develop a new distributed, multiple interacting autonomous agent based traffic signal control architecture to provide effective traffic signal optimization strategies for online optimization of the signal timings for arterial road traffic network.

The objective is also to develop an effective distributed online and batch learning method for optimization of the signal phase timings and rule base adaptation by integrating well-known computational intelligent techniques in the agent decision system. In doing so, this dissertation also seeks to create useful generalized multi-agent systems for solving problems similar to the distributed traffic signal control.

Apart from the objectives related to MAS and traffic signal control, this dissertation also seeks to develop an efficient computational intelligent method of type-reduction to reduce the complexity associated with type-2 fuzzy inference mechanism.

1.3. MAIN CONTRIBUTIONS

The main contributions of this research are in the conceptualization, development and application of a distributed multi-agent architecture to urban traffic signal timing optimization problem. The significant contributions in the design front are as follows.

- The development of a generalized distributed multi-agent framework with hybrid computational intelligent decision making capabilities for homogeneous agent structure.

- The development of deductive reasoning method for the construction of membership functions, rule base of type-2 fuzzy sets and calculating the level of cooperation required between agents.

- The development of cooperation strategies in multi-agent system through internal belief model by incorporating communicated neighbour agent status information.

- The development of symbiotic evolutionary learning method for coevolving membership functions and rule base for the type-2 fuzzy decision system.

- The development of modified Q-learning technique with shared reward values for solving distributed urban traffic signal control problem.

- The development and relocation of the modified type-reducer using neural networks to reduce the computational complexity associated with sorting and defuzzification process in interval type-2 fuzzy sets.

- The development of traffic simulation scenarios to test the reliability and responsiveness of the developed traffic signal controls.

The developed multi agent decision system produced promising results from experiments conducted on simulated road traffic network for different traffic simulation scenarios.

1.4. STRUCTURE OF DISSERTATION

The dissertation consists of eight chapters, and is organized as follows:

Chapter 1 gives a brief introduction of the background on traffic control problem, multi-agent system, the research objectives and the main contributions.

Chapter 2 provides a detailed discussion on distributed multi agent system. It provides a classification of the multi agent system based on the overall agent architecture. The merits and demerits of the various architectures are discussed followed by a description of the communication and coordination techniques used in multi agent systems. It also provides a brief overview of the learning techniques used for evolving the agents to better adapt to the changes in environment.

Chapter 3 describes the various problems associated with urban traffic signal control and some of the promising solution to these problems. A brief overview of the various traffic signal timing optimization methods and their workings are presented. The benchmark traffic signal optimization methods (Hierarchical multi agent system(HMS) and Green link determining system (GLIDE)) used for validating the proposed agent based traffic control system are discussed.

Chapter 4 introduces the proposed distributed multi agent architecture for urban traffic signal timing optimization. The internal structure of the agents and the functionality of each block in an agent are discussed in detail.

Chapter 5 introduces four different types of decision systems used in the proposed multi-agent based traffic signal control. A brief overview of the type-2 fuzzy sets and symbiotic evolutionary genetic algorithm are presented. Design of the decision system based on deductive reasoning, symbiotic evolutionary learning, and Q-learning method is presented in detail. The advantages and disadvantages of the proposed decision systems are highlighted.

Chapter 6 describes in detail, the modelling of a large, complex urban traffic network, Central Business District of Singapore using PARAMICS modeller software. Details of creating the origin-destination matrix used for trip assignment and routing of vehicles inside the simulated road network using the data collected is presented. This chapter provides details of using profile editor to create the traffic release pattern for simulation runs. It also details the performance metrics used to evaluate the performance of the proposed multi-agent systems.

Chapter 7 details the various simulation scenarios used to test the proposed multi agent systems. The travel time delay and speed of vehicles inside the road network for various traffic scenarios using different multi-agent decision control strategies are compared. A detailed analysis of the results and the improvements achieved using proposed signal controls over benchmark traffic controllers are presented.

Chapter 8 concludes the thesis and provides recommendations for future research work.

CHAPTER 2

DISTRIBUTED MULTI-AGENT SYSTEMS

In the previous chapter, a brief introduction of the traffic signal timing optimization problem and suitability of distributed control methods in solving the problem was presented. In order to construct an efficient distributed autonomous multi-agent traffic signal control system with all the required functionalities, it is essential to identify the proper architecture, communication protocol, coordination mechanism and learning to be used.

This chapter provides a detailed review of distributed multi-agent systems, and their architecture, taxonomy, decision making , communication requirements, coordination techniques, and learning methods. This forms the basis for proper design, conceptualisation and implementation of multi-agent systems for real world applications. This chapter also discusses in detail the advantages and disadvantages of various multi-agent architectures, their implementation methodologies, and highlights the significant contributions made by researchers in this field.

2.1 NOTION OF MULTI-AGENT SYSTEMS

Distributed artificial intelligence (DAI) is a subfield of Artificial Intelligence [3] that has gained considerable importance due to its ability to solve complex real-world problems. The primary focus of research in the field of distributed artificial intelligence have been in three different areas. These are parallel AI, Distributed problem solving (DPS) and Multi-agent systems (MAS). Parallel AI primarily refers to methodologies used to facilitate classical AI [4-10]techniques applied to distributed hardware architectures like multiprocessor or cluster based computing.

The main aim of parallel AI is to develop parallel computer architectures, languages and algorithms to increase the speed of operation. Parallel AI is primarily directed towards solving the performance problems of AI systems and not with the conceptual advances in understanding the nature of reasoning and intelligent behaviour among group of agents. Distributed problem solving is similar to parallel AI but considers methodologies of solving a problem by sharing the resources and knowledge between large number of cooperating modules known as "*Computing entity*". In distributed problem solving, communication between computing entities, quantity of information shared are predetermined and embedded in design of the computing entity. Distributed problem solving is rigid due to the embedded strategies and consequently offers little or no flexibility.

In contrast to distributed problem solving, Multi-agent systems (MAS) [11-13] deal with the behaviour of computing entities available to solve a given problem. Multi-agent research is concerned with coordinating intelligent behaviour among all agents– methodology to coordinate the knowledge, goals, skills and plans jointly to solve a problem. In a multi-agent system each computing entity is referred to as an agent. MAS can be defined as a network of individual agents that share knowledge and communicate with each other in order to solve a problem that is beyond the scope of a single agent. It is imperative to understand the characteristics of the individual agent or computing entity to distinguish a simple distributed system from a multi-agent system.

A system with one agent is usually referred to as conventional artificial intelligence technique and a system with multiple agents are called as artificial society. Since distributed systems involve multiple agents, the main issues and the foundations of distributed artificial intelligence are the organisation, co-ordination, and co-operation[14] between the agents.

Multi-agent systems are at the confluence of a wide variety of research disciplines and technologies, notably artificial intelligence, object-oriented programming, human-computer interfaces, and networking[15, 16] . Some of the technologies that have influenced the development of multi-agent systems are as follows

- Database and knowledge-base technology
- Concurrent computing
- Cognitive sciences
- Computational linguistics
- Econometric models
- Biological immune systems

As a result of the existence of such a diversity of contribution, the agents and the multi-agent systems paradigm are diluted in a multitude of perspectives. Researchers in the field of artificial intelligence have so far failed to agree on a consensus definition of the word "*Agent*". The first and foremost reason for this is the universality of the word “*Agent*”. It cannot be owned by a single community. Secondly, the agents can be present in many physical forms, from robots to computer networks. Thirdly, the application domain of the agent is vastly varied and is impossible to generalize. Researchers have used terms like softbots (software agents), knowbots (Knowledge agents), taskbots (task-based agents) based on the application domain where the agents are employed [17]. The most agreed definition of agent is that of Russell and Norvig. They define an agent as a flexible autonomous entity capable of perceiving the environment through the sensors connected to it. The agents act on the environment through actuators. The definition provided does not cover the entire range of characteristics that an agent should possess. Sycara [15] presented some of the most important characteristics that define an agent and are as follows.

- *Situatedness:* This refers to the interaction of an agent with the environment through the use of sensors, and the resultant actions of the actuators. Environment in which an agent is present is an integral part of its design. All of the inputs are received directly as a consequence of the agents interactions with its environment. The agent's directly act upon the environment through the actuators and do not serve merely as a meta level advisor. This attribute differentiates agent systems from expert systems, where the decision making node or entity suggests changes through a middle agent without directly influencing the environment.

- *Autonomy:* This can be defined as the ability of an agent to choose its actions independently without external intervention by other agents in the network (in case of multi-agent systems) or human interference. These attribute protect the internal states of an agent from external influence. It also isolates an agent from instability caused by external disturbances.

- *Inferential capability:* The ability of an agent to work on abstract goal specifications such as deducing an observation by information generalization. This could be done by mining relevant information from the available data.

- *Responsiveness:* The ability to perceive the condition of an environment and respond to it in a timely fashion to take account of any changes in the environment. This latter property is of critical importance in real-time applications.

- *Pro-activeness:* Agents must exhibit a good response to opportunistic behaviour. This is to improve the actions that are goal-directed rather than only being responsive to a specific change in the environment. Agents must have the ability to adapt to any changes in the dynamic environment.

- *Social behaviour:* Even though the agent's decision must be free from external intervention, it must still be able to interact with external sources when the need arises, to achieve a specific goal. It must also be able to share this knowledge and help other agents (MAS) solve a specific problem. That is, agents must be able to learn from the experience of other communicating entities which may be human, other agents in the network, or statistical controllers.

Apart from the above mentioned properties, some of the other important characteristics are mobility, temporal continuity, veracity, collaborative behaviour and rationality. If the agent can satisfy only some of the above mentioned properties like autonomy, social ability, reactivity and pro-activeness, the agent is said to exhibit a weak notion of agency[18].

For an agent to have a strong notion of agency, in addition to the above properties, the agent is required to conceptualise or implement concepts that are more applicable to human like knowledge, belief, intention, obligation or emotion. Another way of giving agents human-like attributes is to represent them visually as animated characters in applications involving human machine interactions. Strong notion of agency tends to be the intersection of all the aspects of different fields that influence the multi-agent systems.

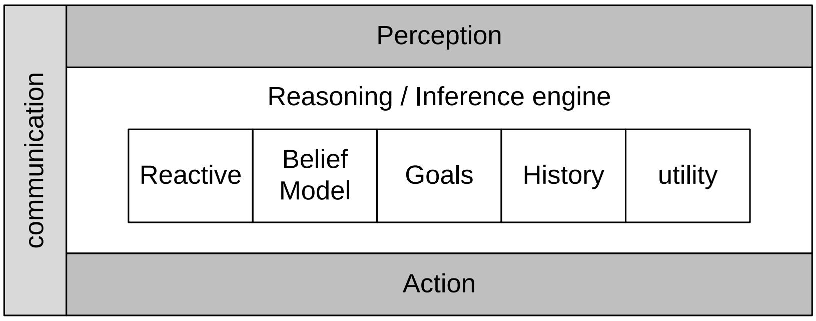


Figure 2.1. Typical building blocks of an autonomous agent

It is however extremely difficult to characterize agents based only on these properties. The characterization of an agent must also be based on the complexity involved in the design, the performed function, and the rationality exhibited. A typical building block of an autonomous agent is shown in Figure 2.1.

2.1.1 Multi-agent System

A Multi-Agent System (MAS) is an extension of the basic agent technology. Definition of multi-agent system can be obtained by the extension of the definition of distributed problem solvers [19] and can be defined as a loosely coupled network of autonomous agents that work together as a society aiming at solving problems that would generally be beyond the problem solving capability of an individual agent. According to [20], the characteristics of a multi-agent systems are:

- Each agent has incomplete information or capabilities for solving the overall problem to be tackled by the system and thus has a very limited viewpoint.

- Lack of global control - The behaviour of the system is influenced by the collective behaviour of individual agents actions and their experiences.

- Decentralization of resources.

Multi-agent systems have been widely adopted in many application domains because of the benefits it offers. Some of the advantages of using MAS technology in large systems [21] are the following:

- An increase in the speed and efficiency of operation due to parallel computation and asynchronous operation.
- A graceful degradation of the system when one or more of the agents fail. It thereby increases the reliability and robustness of the system.
- Scalability and flexibility- Agents can be introduced dynamically into the environment.
- Reduced cost- This is because individual agents cost much less than a centralized architecture.
- Reusability - Agents have a modular structure and hence can be easily reused without major modifications in other systems or upgraded more easily than a monolithic system.

Though multi-agent systems have features that are more beneficial than single agent systems, they also present some critical challenges. Some of the challenges are highlighted in the following section.

- *Environment:* In a multi-agent system, the action of an agent not only modifies its own environment but also that of its neighbours. This necessitates that each agent must predict the action of the other agents, in order to decide the optimal action that would be goal directed. This type of concurrent learning could result in non-stable behaviour and can possibly cause chaos. The problem is further complicated if the environment is dynamic. In such conditions, each agent needs to differentiate between the effects caused due to actions of other agents and variations in the environment.

- *Perception:* In a distributed multi-agent system, the agents are scattered all over the environment. Each agent has a limited sensing capability because of the limited range and coverage of the sensors connected to it. This limits the view available to each of the agents' in the environment. Therefore decisions based on the partial observations made by each of the agents' could be sub-optimal, which in turn affects the global objective.

- *Abstraction:* In agent system, it is assumed that an agent knows its entire action space and mapping of the state space to action space could be performed by the experience gained by each agent. In MAS, every agent does not experience all of the states. To create a map, it must be able to learn from the experience of other agents with similar capabilities or decision making powers. In the case of cooperating agents with similar goals, this can be done by creating communication channel between the agents. In case of competing agents it is not possible to share the information as each of the agent tries to increase its own chance of winning. It is therefore essential to quantify how much of the local information and the capabilities of other agent must be known to create an improved modelling of the environment.

- *Conflict resolution:* Conflicts stem from the lack of global view available to each of the agent. An action selected by an agent to modify a specific internal state may be ineffective for another agent. Under these circumstances, information on the constraints, action preferences and goal priorities of agents must be shared to improve cooperation. A major problem is knowing when to communicate this information and to which of the agents.

- *Inference:* In a single agent system, inference could be easily drawn by mapping the State Space to the Action Space based on trial and error methods. However in MAS, this is difficult as the environment is being modified by multiple agents that may or may not be interacting with each other. Further, MAS may consist of heterogeneous agents, that is agents having different goals and capabilities. Instead of exhibiting a cooperative behaviour, the agents might be competing with each other for a resource. This necessitates identifying a suitable inference mechanism according to the capabilities of each agent to achieve global optimal solution.

It is not necessary to use multi-agent systems for all applications. Some specific application domains which may require interaction with different people or organizations having conflicting or common goals can utilize the advantages presented by MAS in its design.

2.2 CLASSIFICATION OF MULTI-AGENT SYSTEM

Classification of MAS is a difficult task as it can be done based on several different attributes such as Architecture [22], Learning [23-25], Communication [22], Coordination [26]. A general classification encompassing most of these features is shown in Figure 2.2.

2.2.1 Agent Taxonomy

Based on the internal architecture of the individual agents in a multi-agent system, it may be classified into two types:

- Homogeneous structure
- Heterogeneous structure

1) *Homogeneous structure*

In a homogeneous architecture, all agents in the multi-agent system have similar internal architecture. Internal architecture refers to the local goals, sensor capabilities, internal states, inference mechanism and possible action states [27]. The difference between the agents is its physical location and the part of the environment where the specified action is implemented. Each agent receives its inputs from different parts of the environment. There may be overlap in the sensor inputs received. In a typical distributed environment, overlap of sensory inputs is rarely present [28].

2) *Heterogeneous structure*

In a heterogeneous architecture, the agents may differ in their ability, structure, or functionality [29]. Based on the dynamics of the environment and the location of the particular agent, the actions chosen by an agent may differ [30]from the agent located in a different part with the same functionality. Heterogeneous architecture helps in modelling applications much closer to real-world [31]. Each agent can have different local goals that may contradict the objective of other agents. A typical example of this can be seen in the Predator-Prey game [31]. Here both the prey and the predator can be modelled as agents. The objectives of the prey and predator agents are likely to be in direct contradiction to one other.

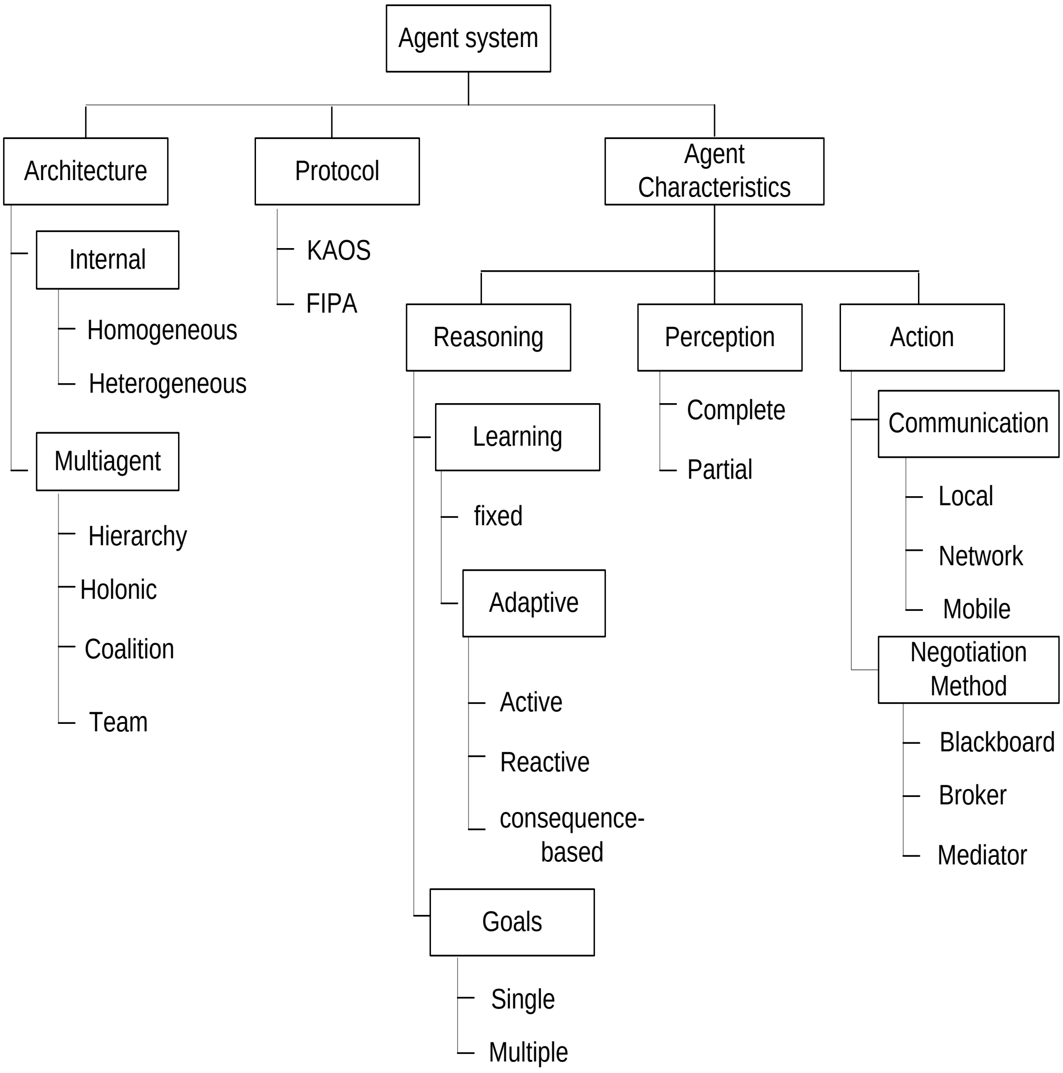


Figure 2.2 . Classification of a multi agent system based on the use of different attributes

2.3 OVERALL AGENT ORGANIZATION

Classification of the multi-agent system based on the organisational paradigm gives a great insight of the strengths and weaknesses of the various types of agent organizations. Based on the organisation structure, the multi-agent system can be classified into four major categories, namely

- Hierarchical

- Holonic
- Coalitions
- Teams

2.3.1 Hierarchical organization

Hierarchical Organization [32] is one of the earliest organizational design in multi-agent systems. Hierarchical architecture has been applied to a large number of distributed problems. In the hierarchical agent architecture, the agents are arranged in a typical tree like structure. The agents at different levels on the tree structure have different levels of autonomy. The data from the lower levels of hierarchy typically flow upwards to agents with a higher hierarchy. The control signal or supervisory signals flow from higher to a lower level of hierarchy [33]. Figure 2.3 shows a typical Three Hierarchical Multi-Agent Architecture. The flow of control signals is from a higher to lower priority agents.

According to the distribution of control between the agents, hierarchical architecture can be further classified as being a simple or uniform hierarchy.

Simple Hierarchy: In a simple hierarchy [34], the decision making authority is bestowed to a single agent at the highest level of the hierarchy. The problem with a simple hierarchy is that a single point failure of the agent in the highest hierarchy may cause the entire system to fail.

Uniform Hierarchy: In a uniform hierarchy, the authority is distributed among the various agents in order to increase the efficiency and fault tolerance in the event of a single or multi-point failures. Decisions are made only by agents with appropriate amount of information. These decisions are sent up the hierarchy only where there is a conflict of interest between agents at different levels of hierarchy.

[33] provides an example of a uniform hierarchical multi-agent system applied to an urban traffic signal control problem. The objective is to provide a distributed control and computation of traffic signal timings. This is to reduce the total delay time experienced by vehicles in a road network. In [32],a three level hierarchical multi-agent system (HMS) was developed. The agents at the lowest level of hierarchy is the intersection agents. Each signal is modelled as an agent and decide their actions autonomously. The zonal agents are one level above the intersections agents in the hierarchy and communicates with a group of intersections. Zonal agents in turn communicate with a central supervisory Regional agent, which occupies the top position in the hierarchy. The intersection decides the optimal green time. This is based on the local information collected at each of the intersections. The agents at the higher level of the hierarchy modify decision of the lower hierarchical agents if there is a conflict of interest or the overall delay experienced by a group of intersections increases due to a selected action. Here, the overall control is uniformly distributed among the agents. Disadvantage with uniform hierarchy, is the amount and the type of information to be transmitted to the agents at higher level of hierarchy. This is a non-trivial problem which gets complicated as the network size increases.

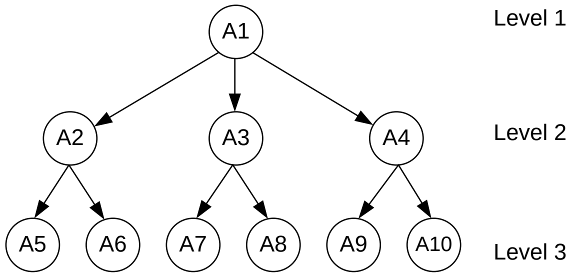


Figure 2.3. A Hierarchical Agent Architecture

2.3.2 Holonic agent organization

A 'Holon' is a stable and coherent or fractal structure that consists of several 'holons' as its sub-structure and is itself a part of a larger framework. The concept of a holon was proposed by Arthur Koestler [35] to explain the social behaviour of biological species. However, the hierarchical structure of the holon and its interactions have been used to model large organizational behaviours in manufacturing and business domains [36-38].

In a holonic multi-agent system, an agent that appears as a single entity may be composed of many sub-agents bound together by commitments. The sub-agents are not bound by any hard constraints or pre-defined rules but through commitments. These refer to the relationships agreed to by all of the participating agents inside the holon.

Each holon appoints or selects a Head Agent that can communicate with the environment or with other agents located in the environment. Selection of the head agent is usually based on the resource availability, communication capability and the internal architecture of each agent. In a homogeneous multi-agent system, the selection can be random and a rotation policy similar to the policy used in distributed wireless sensor networks is employed. In the heterogeneous architecture, head selection is based on the capability of each agent. The holons formed may group further in accordance to benefits foreseen in forming a coherent structure. They form Superholons. Figure 2.4. shows a Superholon formed by grouping two holons. Agents A1 and A4 are the heads of the holons and communicate with agent A7, which is the head of the superholon. The architecture appears to be similar to that of hierarchical organization. However in holonic architecture, cross tree interactions and overlapping group formations are allowed.

The superiority of holonic multi-agent organization and the performance improvements achieved while using holonic group was demonstrated in [38]. The abstraction in the internal working structure of holons provides an increased degree of freedom in selecting the behaviour. A major disadvantage [39] is the lack of a model or knowledge of the internal architecture of the holons. This makes it difficult for other agents to predict the resulting actions of the holons .

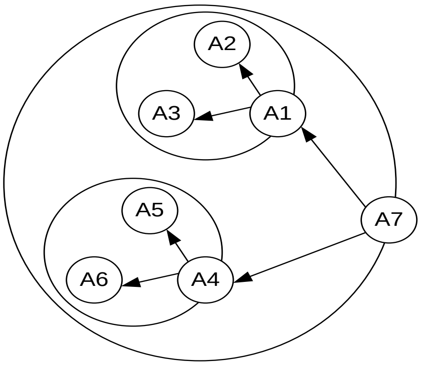


Figure 2.4. An example of Superholon with Nested Holons resembling the Hierarchical MAS

2.3.3 Coalitions

In coalition architecture, a group of agents come together for a short time to increase the utility or performance of the individual agents in a group. The coalition ceases to exist when the performance goal is achieved. Figure 2.5. shows a typical coalition multi-agent system. The agents forming the coalition may have either a uniform or a hierarchical architecture. Even when using a uniform architecture, it is possible to have a leading agent to act as a representative of the coalition group. The overlap of agents among coalition groups is allowed as this increases the common knowledge within the coalition group and helps in the construction of belief model. However the presence of overlapping agents increases the complexity in computation of the negotiation strategy. Coalition architecture is difficult to maintain in a dynamic environment due to the shift in the performance of coalition group. It may be necessary to regroup agents in order to maximize the system performance.

Theoretically, forming a single group consisting of all the agents in the environment will maximize the performance of the system. This is because each agent has access to all the information and resources necessary to calculate the conditions for optimal action. It is practically not feasible to form such a coalition due to constraints on the communication and resources.

The number of coalition groups created must be minimized in order to reduce the cost associated with creating and dissolving a coalition group. The group formation may be pre-defined based on a threshold set for performance measure or alternatively could be evolved online.

In [40], a coalition multi-agent architecture for urban traffic signal control was developed. Each intersection was modelled as an agent with capability to decide the optimal green time required for that intersection. A distributed neuro-fuzzy inference engine was used to compute the level of cooperation required and the members of the dynamically formed coalition group.

The coalition groups reorganize and regroup dynamically with respect to the changing traffic input pattern. Disadvantage is the increased computational complexity involved in creating ensembles or coalition groups. The coalition MAS may have a better short term performance than other agent architectures [41].

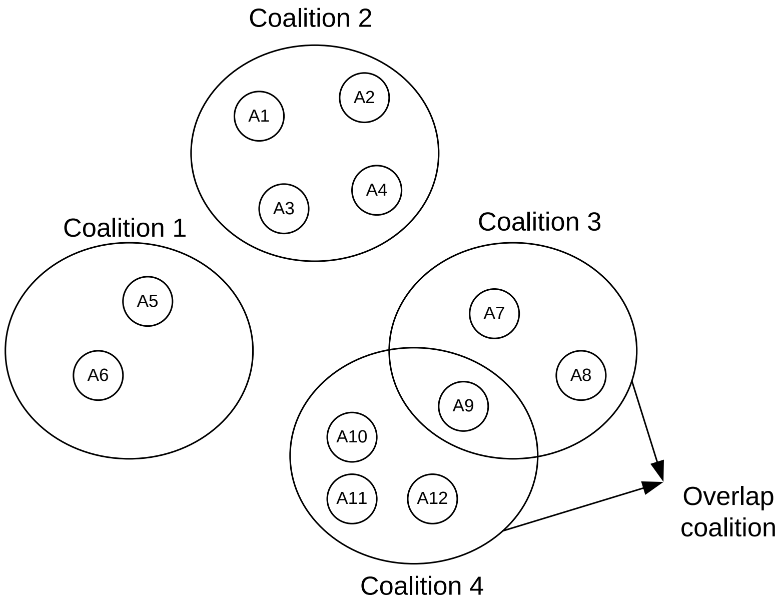


Figure 2.5. Coalition multi agent architecture using overlapping groups

2.3.4 Teams

Team MAS architecture [42] is similar to coalition architecture in design except that the agents in a team work together to increase the overall performance of the group rather than each working as individual agents. The interactions between agents within a team can be quite arbitrary, and the goals or the roles assigned to each of the agents can vary with time based on improvements resulting from the team performance. Reference [43] deals with a team based multi-agent architecture having a partially observable environment. In other words, teams that cannot communicate with each other has been proposed for the Arthur's bar problem. Each team decides on whether to attend a bar or not by means of predictions based on the previous behavioural pattern and the crowd level experienced which is the reward or the utility received associated with the specific period of time. Based on the observations made in [43], it can be concluded that a large team size is not beneficial under all conditions.

Consequently some compromise must be made between the amount of information, number of agents in the team and the learning capabilities of the agents.

Large teams offer better visibility of the environment and larger amount of relevant information. However, learning or incorporating the experiences of individual agents into a single framework team is affected. A smaller team size offers faster learning possibilities but result in sub-optimal performance due to a limited view of the environment. Tradeoffs between learning and performance need to be made in the selection of the optimal team size. This increases the computational cost much greater than that experienced in coalition multi-agent system architecture. Figure 2.6. shows a typical team based on architecture with partial view. The team 1 and 3 can see each other but not teams 2 ,4 and vice versa. The internal behaviour of the agents and their roles are arbitrary and vary with teams even in homogeneous agent structure.

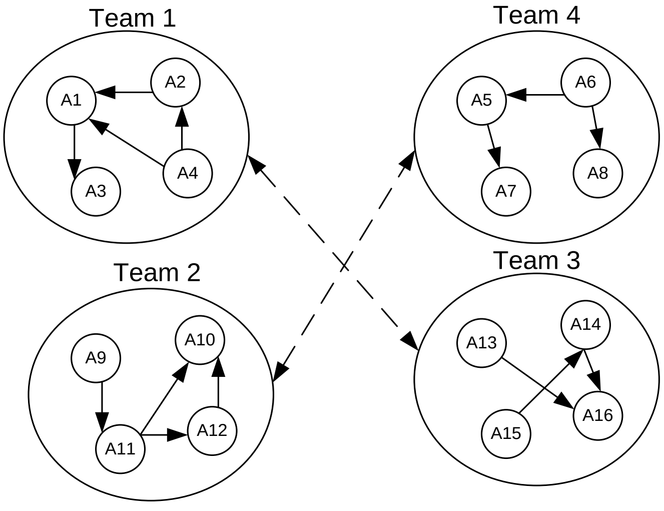


Figure 2.6. Team based multi agent architecture with a partial view of the other teams

Variations and constraints on aspects of the four agent architecture mentioned earlier in this chapter can form other architectures such as federations, societies and congregations. Most of these architectures are inspired by behavioural patterns in governments, institutions and large industrial organizations. A detailed description of these architectures, their formation and characteristics may be found in [42].

2.4 COMMUNICATION IN MULTI-AGENT SYSTEMS

Communication is one of the crucial components in multi-agent systems that needs careful consideration. Unnecessary or redundant intra-agent communication can increase the cost and cause instability. Communication in a multi-agent system can be classified into two types. This is based on the architecture of the agent system and the type of information communicated between the agents. In [22], the various issues with homogeneous and heterogeneous MAS architecture has been described and demonstrated using a predator/prey and robotic soccer games. On the basis of the information communication between the agents [44], MAS can classified as local communication or message passing and network communication or Blackboard. Mobile communication can be categorized into class of local communication.

2.4.1 Local communication

Local communication lack memory and intermediate communication media to store information and act as a facilitator respectively. The term message passing is used to emphasize the direct communication between the agents. Figure 2.7. shows the structure of the message passing communication between agents. In this type of communication, the information flow is bidirectional. It creates a distributed architecture and reduces the bottleneck caused by failure of central agents. This type of communication has been used in [33, 45, 46].

2.4.2 Blackboards

Another way of exchanging information between agents is through Blackboards [47]. Agent-based blackboards, like federation systems, use grouping to manage the interactions between agents. In blackboard communication, a group of agents share a data repository used for efficient storage and retrieval of data actively shared between the agents. The repository can hold both the design data as well as the control knowledge and are accessible to the agents. The type of data that can be accessed by an agent can be controlled through the use of a control shell. This acts as a network interface that notifies the agent when relevant data is available in the repository. The control shell can be programmed to establish different types of coordination among the agents. Neither the agent groups nor the individual agents in the group need to be physically located near the blackboards. It is possible to establish communication between various groups by remote interface communication. The major issue is the loss of critical information due to the failure of blackboards. This could render the group of agents useless depending on the information stored in the specific blackboard. However, it is possible to establish some redundancy and share resources between various blackboards. Figure 2.8a. shows a single blackboard with the group of agents associated with it. Figure 2.8b. shows blackboard communication between two different agent groups and also the location of facilitator agents in each group.

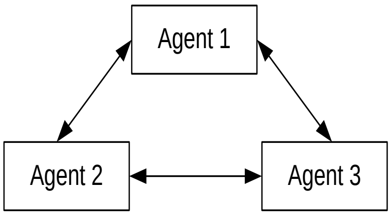


Figure 2.7. Message Passing Communication between agents

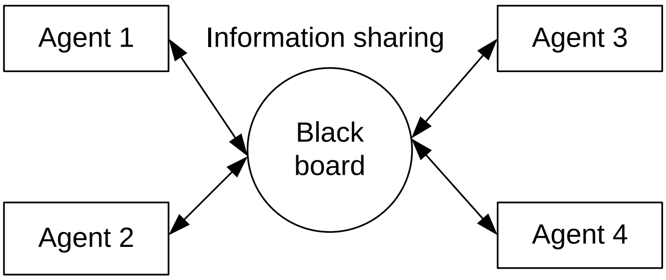


Figure 2.8a. Blackboard type communication between agents

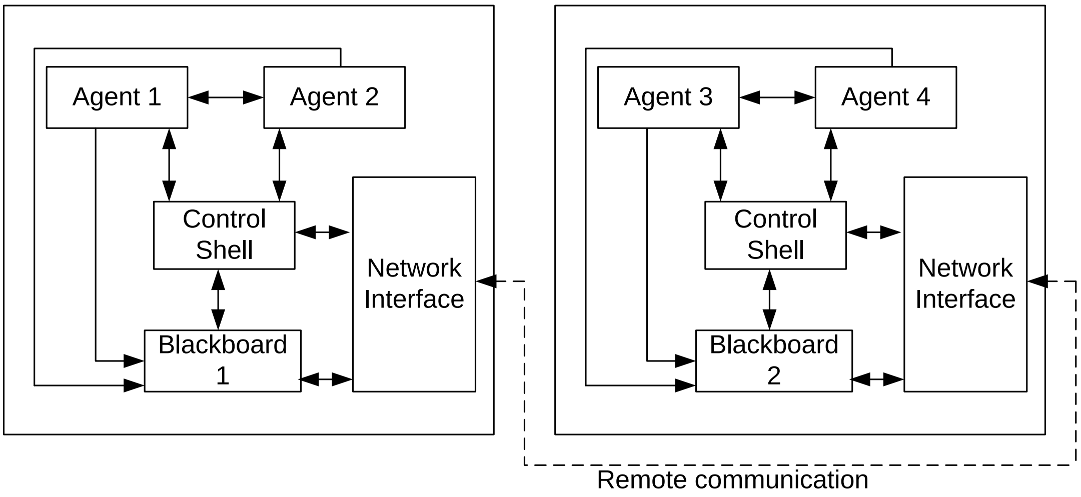


Figure 2.8b. Blackboard communication using remote communication between agent groups

2.4.3 Agent Communication Language

An increase in the number of agents and the heterogeneity of the group necessitates a common framework to help in proper interaction and information sharing. This common framework is provided by the agent communication languages (ACL). The elements that are of prime importance in the design of ACL were highlighted in [48, 49].

Agent common language provides the necessary interaction format (Protocol) that can be understood by all of the participating agents. The communication language also provides a shared ontology where the message communicated has the same meaning in all contexts and allows agent independent semantics. In order to perform the task of agents effectively, agents depend heavily on expressive communication with other agents to perform the requests, to propagate the information capabilities, and to negotiate with other agents. Designing a proper communication language has two major problems.

- *Inconsistencies* in the use of syntax or vocabulary. Same words could have entirely different or even conflicting meanings with respect to different agents
- *Incompatibilities* between different programs using different words or expressions to convey the same information.

There are two popular approaches in the design of an agent communication language, procedural approach and declarative approach. In Procedural approach, the communication between the agents is modelled as sharing of procedural directives. Procedural directives could be task specific working instructions or general working mechanism of the agent. Scripting languages are commonly used in the procedural approach. Some of the most common scripting languages employed are JAVA, TCL, Applescript and Telescript.

The major disadvantage of procedural approach is the necessity of providing information on the recipient agent, which in most cases is not known or only partially known. In case of making a wrong model assumption, the procedural approach may have a destructive effect on the performance of the agents. The second major concern is with the merging of shared procedural scripts into a relevant single large executable script for the agent. Owing to these disadvantages, the procedural approach is not the preferred method for designing agent communication language.

In the declarative approach, the agent communication language is designed and based on the sharing of declarative statements that specifies definitions, assumptions, assertions, axioms etc. For the proper design of an ACL using a declarative approach, the declarative statements must be sufficiently expressive to encompass the use of a wide-variety of information. This would increase the scope of the agent system and also avoid the necessity of using specialized methods to pass certain functions. The declarative statements must be short and precise. An increase in the length of the declarative statements affects the cost of communication between agents and also increases the probability of information corruption. The declarative statements also needs to be simple enough to avoid the use of a high level language to code. To meet all the above requirements of the declarative approach based ACL, the ARPA knowledge sharing effort had devised an agent communication language.

The ACL designed consists of three parts [49]: A Vocabulary part, "Inner language" and "Outer language". The Inner language is responsible for the translation of the communication information into a logical form that is understood by all agents. There is still no consensus on a single language and many inner language representations like KIF (Knowledge Interchange Format)[50], KRSL, LOOM are available. The linguistic representation created by these inner languages are concise, unambiguous and context-dependent. The receivers must derive from them the original logical form. For each linguistic representation, ACL maintains a large vocabulary repository. A good ACL maintains this repository open-ended so that modifications and additions can be made to include increased functionality. The repository must also maintain multiple ontology's and its usage will depends on the application domain.

Knowledge Interchange Format [51] is one of the best known inner languages and it is an extension of the First-Order Predicate Calculus (FOPC). Some of the information that can be encoded using KIF are simple data, constraints, negations, disjunctions, rules, meta-level information that aids in the final decision process. It is not possible to use just the KIF for information exchange as much implicit information needs to be embedded. This is so that the receiving agent can interpret it with a minimal knowledge of the sender's structure. This is difficult to achieve as the packet size grows with the increase in embedded information. To overcome this bottleneck, a high level language that utilizes the inner language as its backbone were introduced. These high-level languages make the information exchange independent of the content syntax and ontology. One well known Outer language that satisfies this category is the KQML (Knowledge Query and Manipulation Language) [52]. A typical information exchange between two agents utilizing the KQML and KIF agent communication language is as follows.

```
  (ask  :Content (geolocation lax(?long ?lat))
        : language KIF
        :ontology STD_GEO
       : from location_agent
      : to  location_server
     : label Query- "Query identifier")
(tell : content "geolocation(lax, [55.33,45.56])"
     : language standard_prolog
```

: ontology STD_GEO)

The KQML is conceived as both message format and message handling protocol to facilitate smooth communication between agents. From the above example provided, it can be seen that KQML consists of three layers (Figure 2.9): A communication layer which indicates the origin and destination agent information and query label or identifier, a message layer that specifies the function to be performed (eg: In the example provided, the first agent asks for the geographic location and the second agent replies to the query), and a content layer to provide the necessary details to perform the specific query.

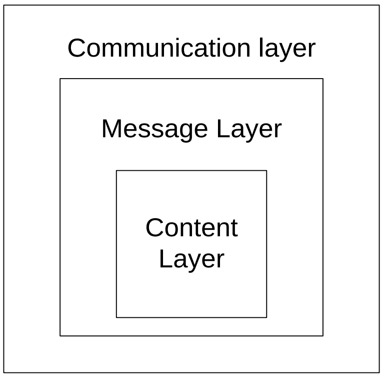


Figure 2.9. KQML - Layered language structure

In KQML, the communication layer is at a low level and packet oriented. A stream oriented approach is yet to be developed. The communication streams could be built on TCP/IP, RDP, UDP or any other packet communication media. The content layer specifies the language to be employed by the agent. It should be noted that agents can use different languages to communicate with each other and interpretation can be performed locally using higher level languages.

2.5 DECISION MAKING IN MULTI-AGENT SYSTEM

Multi-agent decision making is different from a simple single agent decision system. The uncertainty associated with the effects of a specific action on the environment and the dynamic variation in the environment as a result of the action of other agents makes multi-agent decision making a difficult task. Usually the decision making in MAS is considered as a methodology to find a joint action or the equilibrium point which maximizes the reward received by every agent participating in decision making process. The decision making in MAS can be typically modelled as a game theoretic method. Strategic game is the most simplest form of decision making process. Here every agent chooses its actions at the beginning of the game and the simultaneous execution of the chosen action by all agents.

A strategic game [53] consists of a set of players - in multi-agent scenario, the agents are assumed to be the players.

- For each player, there is a set of actions
- For each player, the preferences over a set of action profiles

There is a payoff associated with each of the combination of action values for the participating players. The payoff function is assumed to be predefined and known in the case of a simple strategic game. It is also assumed that the actions of all agents are observable and is a common knowledge available to all agents. A solution to a specific game is the prediction of the outcome of the game making the assumption that all participating agents are rational.

The prisoner's dilemma is a best case for demonstrating the application of game theory in decision making involving multiple agents. The prisoner's dilemma problem can be states as :

"*Two suspects involved in the same crime are interrogated independently. If both the prisoner's confess to the crime, each of them will end up spending three years in prison. If only one of the prisoner confesses to the crime, the confessor is free while the other person will spend four years in prison. If they both do not confess to the crime, each will spend a year in prison*."

This scenario can be represented as a strategic game.

- *Players* Two suspects involved in the crime
- *Actions* Each agent's set of actions is {*Not confess, confess*}
- *Preferences* Ordering of the action profile for agent 1, from best to worst case scenario, is {*confess, Not confess*}, {*Not Confess, Not confess*}, {*Confess, Confess*} and {*Not confess, Confess*}. Similar ordering could be performed by agent 2.

A payoff matrix that represents the particular preferences of the agents needs to be created. Simple payoff matrix can be

u1{Confess, Not confess} =3,

u1{Not confess, Not confess}=2.

u1{Confess, Confess}=1,

u1{Not confess, confess}=0.

Similarly the utility or payoff for agent 2 can be represented as

u2{Not confess, confess}=3,

u2(Not confess, Not confess}=2,

u2{confess, Not confess}=0

u2{confess, confess}=1.

The reward or payoff received by each agent for choosing a specific joint action can be represented in a matrix format called as payoff matrix table. The problem depicts a scenario where the agents can gain if they cooperate with each other but there is also a possibility to be free if a confession is made. The particular problem can be represented as a payoff matrix as shown in Figure 2.10. In this case it can be seen that the solution "*Not confess"* is strictly dominating. By strictly dominating solution, it means that a specific action of an agent always increases the payoff of the agent irrespective of the other agents actions.

		Agent 1	
		Not Confess	Confess
Agent 2	Not Confess	2,2	0,3
	Confess	3,0	1,1

Figure 2.10. Payoff matrix for the Prisoner's Dilemma Problem

However, there can be variations to the prisoner's dilemma problem by introducing an altruistic preference while still calculating the payoff of the actions. Under this circumstance, there is no action strictly dominated by the other.

2.5.1 Nash equilibrium

To obtain the best solution based on the constructed payoff matrix, the most common method employed is the Nash Equilibrium [54], which can be stated as follows:

A Nash Equilibrium is an action profile a with the property that no player i can do better by choosing an action different from a* of i, given that every other player adheres to a* of j.*

In the most idealistic conditions, where the components of the game are drawn randomly from a collection of populations or agents, a Nash equilibrium corresponds to a steady state value. In a strategic game, there always exists a Nash equilibrium but it is not necessarily a unique solution. Examining the payoff matrix in Figure 2.11 shows that {*confess, confess*} is the Nash equilibrium for the particular problem. The action pair {*confess, confess*} is a Nash equilibrium because given that agent 2 chooses to confess, agent 1 is better off choosing *confess* than *Non confess*. By a similar argument with respect to agent 2 it can be concluded that {*confess, confess*} is a Nash Equilibrium. In particular, the incentive to have a free ride on confession eliminates any possibility of selecting mutually desirable outcome of the type {*Not Confess, Not Confess*}. If the payoff matrix could be modified to add value based on the trust or reward to create altruistic behaviour and feeling of indignation, then the subtle balance that exists shifts and the problem would have a multiple number of Nash equilibrium points as shown in Fig.2.11.

		Agent 1	
		Not Confess	Confess
Agent 2	Not Confess	2,2	-2,-1
	Confess	-1,-2	1,1

Figure 2.11. Modified Payoff matrix for the Prisoner's Dilemma Problem

In modified prisoner’s dilemma problem, a single action does not always dominate and multiple equilibrium points exist. To obtain a solution for this type of problem, coordination between the agents is an essential requirement.

2.5.2 The Iterated elimination method

The solution to the Prisoner's dilemma problem can also be obtained by using the iterated elimination method [55]. In this method, the strongly dominating actions are iteratively eliminated until no more actions are strictly dominating. The iterated elimination method assumes that all agents exhibit a rational behaviour and will not choose a strictly dominant solution. This method is weaker than the Nash equilibrium as it finds the solution by means of a algorithm. Iterated elimination method fails when there are no strictly dominant actions available in the solution space. This limits the applicability of the method in multi-agent scenario where mostly weakly-dominant actions are encountered.

2.6 COORDINATION IN MULTI-AGENT SYSTEM

Coordination is the central issue in the design of multi-agent systems. Agents are seldom stand-alone systems and usually involve more than one agent working in parallel to achieve a common goal. When multiple agents are employed to achieve a goal, it is necessary to coordinate or synchronize their actions to ensure the stability of the system. Coordination between agents increases the chances of attaining a optimal global solution. In [56], major reasons necessitating coordination between the agents were highlighted. The requirements are

- To prevent chaos and anarchy
- To meet global constraints
- To utilize distributed resources, expertise and information
- To prevent conflicts between agents
- To improve the overall efficiency of the system

Coordination can be achieved by applying constraints on the joint action choices of each agent or by utilizing the information collated from neighbouring agents. These are used to compute the equilibrium action point that could effectively enhance the utility of all the participating agents. Applying constraints on the joint actions requires an extensive knowledge of the application domain. This may not be readily available. It necessitates the selection of the proper action taken by each agent based on the equilibrium point computed. However, the payoff matrix necessary to compute the utility value of all action choices might be difficult to determine. The dimension of the payoff matrix grows exponentially with an increase in the number of agents and the available action choices. This may create a bottleneck when computing the optimal solution.

The problem of dimensional explosion can be solved by dividing the game into a number of sub-games that can be more effectively solved. A simple mechanism which can reduce the number of action choices is to apply constraints or assign roles to each agent. Once a specific role is assigned, the number of permitted action choices is reduced and simplifies the computation of payoff matrix. This approach is of particular importance in a distributed coordination mechanism. However, in centralized coordination techniques this is not a major concern as it is possible to construct belief models for all agents. The payoff matrix can be computed centrally and communicated to all of the agents as a shared resource.

Centralized coordination is adopted from the basic client/server model of coordination. Most of the centralized coordination techniques uses blackboards as a way in which to exchange information. Master agent schedules of all the connected agents are required to read and write information from and to the central information repository. Some of the commonly adopted client/server models are KASBAH[57] and MAGMA[58]. The model uses a global blackboard to achieve the required coordination. Disadvantage in using the centralized coordination is the disintegration of the system due to single point failure of the repository or the mediating agent. Further, use of the centralized coordination technique is contradictory to the basic assumption of DAI[56].

2.6.1 Coordination through protocol

A classic coordination technique among agents in a distributed architecture is through the communication protocol. Protocol is usually written in high level language and specifies the method of coordination between the agents as a series of task and resource allocation methods. The most widely used protocol is the Contract Net Protocol [59] which facilitates the use of distributed control for cooperative task execution. The protocol specifies the information to be communicated between the agents and the format of information dissemination. A low-level communication language such as KIF that can handle the communication streams is assumed to be available. The protocol engages in negotiation between the agents to arrive at an appropriate solution. The negotiation process must adhere to the following characteristics

- Negotiation is a local process between agents and it involves no central control
- Two way communication between all participating agents exists
- Each agent makes its evaluation based on its own perception of the environment
- The final agreement is made through a mutual selection of the action plan

Each agent assumes the role of *Manager* and *Contractor* as necessary. The manager essentially serves to break a larger problem into smaller sub-problems and finds contractors that can perform these functions effectively. A contractor can become a manager and decompose the sub-problem so as to reduce the computational cost and increase efficiency. The manager contracts with a contractor through a process of bidding. In the bidding process, the manager specifies the type of resource required and a description of the problem to be solved. Agents that are free or idle and have the resources required to perform the operation submits a bid indicating their capabilities. The manager agent then evaluates the received bids, chooses an appropriate contractor agent and awards the contract. In case of non-availability of any suitable contracting agent, the manager agent waits for a pre-specified period before rebroadcasting the contract to all agents. The contracting agent may negotiate with the manager agent seeking an access to a particular resource as a condition before accepting the contract.

The FIPA model [60] is the best example of an agent platform that utilizes the contract net protocol to achieve coordination in between the agents. FIPA - Foundation for Intelligent Physical Agents is a model developed to standardize agent technology. The FIPA has its own ACL (Agent Communication Language) that serves as the backbone for the high-level contract net protocol.

Disadvantage of the protocol based coordination is the assumption of the existence of a cooperative agents. The negotiation strategy is passive and does not involve any punitive measures to force an agent to adopt a specific strategy. Usually a common strategy is achieved through iterative communication, where the negotiation parameters are modified progressively to achieve equilibrium. This makes the contract net protocol to be communication intensive.

2.6.2 Coordination via graphs

Coordination graphs were introduced in [61] to serve as a framework to solve large scale distributed coordination problems. In coordination graphs, each problem is sub-divided into smaller problems that are easier to solve. The main assumption with coordination graphs is that the payoffs can be expressed as a linear combination of the local payoffs of the sub-game. Based on this assumption, algorithm such as variable elimination method can compute the optimal joint actions by iteratively eliminating agents and creating new conditional functions that compute the maximal value the agent can achieve given the action of other agents on which it depends. The joint action choice is only known after the completion of the entire computation process, which scales with the increase in agents and available action choices and is of concern in time critical processes. An alternate method using max-plus which reduces the computation time required was used in [58]. This was to achieve coordination in multi-agent system when applied to urban traffic signal control.

2.6.3 Coordination through belief models

In scenarios where time is of critical importance, coordination through protocols fail to succeed when an agent with a specific resource to solve the sub-problem rejects the bid. In such scenarios, agents with an internal belief model of the neighbouring agents could solve the problem. The internal belief model could be either evolved by observing the variation in the dynamics of the environment or developed based on heuristic knowledge and domain expertise. When the internal model is evolved, the agent has to be intelligent enough to differentiate between the change in its environment due to the actions of other agents and natural variations occurring in the environment. In [28], a heuristics based belief model has been employed to create coordination between agents and to effectively change the green time. In [62], evolutionary methods combined with neural networks have been employed to dynamically compute the level of cooperation required between the agents. This is based on the internal state model of the agents. The internal state model was updated using reinforcement learning methods. A disadvantage using the coordination based on belief model for the agents is an incorrect model could cause chaos due to the actions selected.

2.7 LEARNING IN MULTI-AGENT SYSTEM

The learning of an agent can be defined as building or modifying the belief structure based on the knowledge base, input information available and the consequences or actions needed to achieve the local goal [63]. Based on the above definition, agent learning can be classified into three types.

- Active learning
- Reactive learning
- Learning based on consequence

In active and reactive learning, the update of the belief part of the agent is given preference over an optimal action selection strategy as a better belief model could increase the probability of the selection of an appropriate action.

2.7.1 Active learning

Active learning can be described as a process of analysing the observations to create a belief or internal model of the corresponding situated agent's environment. The active learning process can be performed by using a deductive, inductive or probabilistic reasoning approach.

In the deductive learning approach, the agent draws a deductive inference to explain a particular instance or state-action sequence using its knowledge base. Since the result learned is implied or deduced from the original knowledge base which already exists, the information learnt by each agent is not a new but useful inference. The local goal of each agent could form a part of the knowledge base. In the deductive learning approach, the uncertainty or the inconsistency associated with the agent knowledge is usually disregarded. This makes it not suitable for real-time applications.

In inductive learning approach, the agent learns from observations of state-action pair. These viewed as the instantiation of some underlying general rules or theories without the aid of a teacher or a reference model. Inductive learning is effective when the environment can be presented in terms of some generalized statements. Well known inductive learning approaches utilize the correlation between the observations and the final action space to create the internal state model of the agent. The functionality of inductive learning may be enhanced if the knowledge base is used as a supplement to infer the state model. The inductive learning approach suffers at the beginning of operation as statistically significant data pertaining to the agent may not be available.

The probabilistic learning approach is based on the assumption that the agent knowledge base or the belief model can be represented as probabilities of occurrence of events. The agent's observation of the environment is used to predict the internal state of the agent. One of the best examples of probabilistic learning is that of the Bayesian theorem. According to the Bayesian theorem, the posterior probability of an event can be determined by the prior probability of that event and the likelihood of its occurrence. The likelihood probability can be calculated based on observations of the samples collected from the environment and prior probability can be updated using the posterior probability calculated in the previous time step of the learning process. In a multi-agent scenario where the action of one agent influences the state of other agent, the application of using the probabilistic learning approach is difficult. This stems from the major knowledge requirement of the joint probability of actions and state space of different agents. With an increase in the number of agents, it is difficult in practice to calculate and infeasible computationally. The other limitation is the limited number of the sample observations available to estimate the correct trajectory.

2.7.2 Reactive learning

The process of updating a belief without having the actual knowledge of what needs to be learnt or observed is called Reactive Learning. This method is particularly useful when the underlying model of the agent or the environment is not clearly known. Reactive learning can be seen in agents which utilize connectionist systems such as neural networks. Neural networks depend on the mechanism which maps the inputs to output data samples using inter-connected computational layers. Learning is performed by adjustment of synaptic weights between the layers. In [64], reactive multi-agent based feed forward neural networks have been used and its application to the identification of non-linear dynamic system have been demonstrated. In [65] many other reactive learning methods such as accidental learning, go-with-the-flow, channel multiplexing and a shopping around approach have been discussed. Most of these methods are rarely employed in a real application environment as most of them are designed exclusively to a specific application domain.

2.7.3 Learning based on consequences

Learning methods presented in the previous sections were concerned with understanding the environment based on the belief model update and analysis of patterns in sample observations. This section will deal with the learning methods based on the evaluation of the goodness of selected action. This may be performed by reinforcement learning methods.

Reinforcement learning is a way of programming the agents using reward and punishment scalar signals without specifying how the task is to be achieved. In reinforcement learning, the behaviour of the agent is learnt through trial and error interaction with the dynamic environment without an external teacher or supervisor that knows the right solution. Conventionally, reinforcement learning methods are used when the action space is small and discrete. Recent developments in reinforcement learning techniques have made it possible to use the methods in continuous and large state-action space scenarios too. Examples of applications using reinforcement learning techniques in reactive agents are given in [66, 67].

In reinforcement learning [68], the agent attempts to maximize the discounted scalar reward received from the environment over a finite period of time. To represent this, an agent is represented as a Markov Decision Process.

- A discrete number of states $s \in S$
- A discrete set of actions $a \in A$
- State transition probability $p(s'|s,a)$
- Reward function $R : SXA \rightarrow °$

The reward function can be written as $R = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$. The objective is to maximize this function for a given policy function. A policy is a mapping from the state to the action values. The optimal value of a specific state *s* can be defined as the maximum discounted future reward which is received by following a specific stationary policy and can be written as

$$V*(s) = \max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_t = \pi(s)\right] \tag{2.1}$$

The expectation operator averages the transition values. In a similar manner the Q value can be written as

$$Q*(s,a) = \max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a\right] \tag{2.2}$$

The optimal policy can then be determined as *arg max* of the Q-value. To compute the optimal value function and the Q-value, the Bellman equation (2.3) and (2.4) is used. The solution to Bellman equation can be obtained by recursively computing the values using dynamic programming techniques. However, the transition probability values are difficult to obtain. Therefore the solution is obtained iteratively by using the temporal difference error between the value of successive iterations as shown in (2.5) and (2.6).

$$V^*(s) = \max_{\pi} \left[R(s,a) + \gamma \sum_{s'} p(s' \mid s,a) V^*(s') \right] \qquad (2.3)$$

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} p(s' \mid s,a) V(s') \qquad (2.4)$$

$$V(s) \leftarrow V(s) + \alpha \left[r + \gamma V(s') - V(s) \right] \qquad (2.5)$$

$$Q(s_t, a_t) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \qquad (2.6)$$

The solution to (6) is referred to as the q-learning method. The Q-value computed for each state action pair is stored in Q-map and is used to update the Q-values. Based on the Q-values, the appropriate actions are selected. The major disadvantage is that the exploration and exploitation trade-off must be determined. To build an efficient Q-map, it is essential to compute the Q-values corresponding to all the state-action pair. The convergence is guaranteed if all the state-action pairs have been visited infinite number of times (theoretical).

In single agent RL, the convergence and methodologies are well defined and proven. In a distributed MAS, the reinforcement learning method faces the problem of combinatorial explosion with increase in the state and action space. Another major concern is that the information must be passed between the agents for effective learning. In [69], distributed value function based on RL has been described. The value functions are shared among the agents thereby increasing the global view available to each agent. For effective distributed learning through shared value function, all agents must have similar architecture and the inputs and state space are similar for all agents. A complete survey of reinforcement learning can be found in [70] .

2.8 SUMMARY

In this chapter, a detailed survey of the existing agent architectures, taxonomy, communication requirements, coordination mechanism, decision making and learning in multi-agent systems applied to wide range of applications have been presented. The insights derived from the survey of the existing design methodology will be useful in conceptualizing and implementing an effective distributed multi-agent system for complex urban traffic signal control problem.

CHAPTER 3

REVIEW OF ADVANCED SIGNAL CONTROL TECHNIQUES

In order to construct an efficient distributed multi-agent based traffic signal control system, it is essential to review the existing traffic control methods, their advantages and disadvantages. This chapter presents a classification of the existing traffic signal control methods and provides a detailed review of the working of various existing traffic control methods. The chapter also details the working mechanism of the benchmark traffic signal controls HMS and GLIDE used for evaluating the performance of the proposed multi-agent based traffic signal controls.

3.1 CLASSIFICATION OF TRAFFIC SIGNAL CONTROL METHODS

Traffic signal controls can be classified into three types based on the type of architecture used to obtain the green time required for each phase in a cycle.

- Fixed-time control
- Traffic Actuated control
- Traffic Adaptive control

3.1.1 Fixed-time control

In fixed-time control, the duration and the order of all green phases are pre-fixed. Fixed time control assumes that the traffic patterns can be predicted accurately based on historical data. The traffic situation changes over time and usually employs a clock to replace one fixed-time control plan with another. As fixed-time controllers do not require traffic detectors to be installed at the intersection, the construction cost is much lower than with traffic actuated and traffic-adaptive control. The main drawback of fixed-time control is that it is not capable of adapting itself to the real time traffic patterns as it is based only on historical data. Historical data is often not representative of the current situation as:

- Traffic arrives at the intersection randomly, which makes it impossible to predict the traffic demand accurately.

- Traffic demand changes and shifts over a long period of time leading to “aging” of the optimized settings.

- Traffic demand may change due to drivers’ response to the new optimized signal settings.

- Events, accidents, and other disturbances may disrupt traffic conditions in a non-predictable way.

In fixed-time control, the signal cycle is divided over the various phases according to historical traffic volumes. As a consequence of the time needed to clear the intersection when changing phases and for traffic to start-up, a fixed amount of time during the signal cycle can be considered lost and cannot be effectively used for traffic flow. The amount of time lost (per hour) increases when the duration of the signal cycle is chosen shorter. Intersections with a shorter signal cycle therefore would have a lower overall capacity. However, longer signal cycles also lead to longer waiting times and longer queues leading to saturation of the links. This necessitates fixing the maximum and minimum bounds for green time and cycle length. In order to find an optimal value for the cycle and the green durations of the separate phases, Webster had derived the formula utilizing the flow rate experienced at each lane of the link. Based on how the green time is computed, the fixed-time control can be classified into two types:

- *Progression based methods* : Maximizes the bandwidth of the progression – PASSER (Progression Analysis and Signal System Evaluation Routine)[71], MAXBAND[72] and MULTIBAND[73]

- *Disutility based methods* : Minimizes a performance measures like overall travel time delay and number of stops – TRANSYT-7F[74] and SYNCHRO[75]

3.1.2 Traffic actuated control

For vehicle-actuated and traffic-actuated control, detectors are needed to get information about the actual traffic situation. The detectors that are used most frequently in practice are inductive loop detectors. In order to decide whether it is efficient to terminate the green phase, the traffic-actuated controller should be able to determine whether the last vehicle of the queue that has build up at the stop line during the red phase has passed. This is performed by measuring the gap between the vehicles. If the gap between vehicles is larger than a threshold maximum gap, the control program decides to terminate the green phase. Additionally many traffic-actuated controllers also extend the green time to ensure that the green phase is terminated safely and in a comfortable manner.

These extensions continue until the intervals between vehicles are long enough for the signals to decide that it would be more efficient to terminate the current green phase or until a pre-specified maximum green time has been reached. There are four major regions that needs to be monitored - Zone 1, Zone 2, Option zone and Comfort zone. Zone 1 and 2 are very close to the stop line intersection. Zone 1 is a 3m region close to the stop line intersection followed by Zone 2 which extends for a further 20m beyond the first vehicle after the Zone 1. Option zone is the overlapping region extending beyond zone 2. If the vehicles are present within this region, the green time of the phase can be effectively terminated. In this region, the traffic flow is considerably slower and varies with traffic conditions. Comfort zone extends beyond the option zone where the vehicle flow is steady and is not influenced much unless there is a queue build-up. Presence of vehicles in any of these regions can cause the extension of the green time. However, the priority is different in each region.

Coordination between traffic-actuated controllers is achieved on the basis of the same principles through which coordination between fixed-time controllers is achieved. In order to ensure that traffic-actuated controllers return to the coordinated phase in time a mechanism must be in place to force non-coordinated phases to terminate. Two types of force-off modes are used : floating and fixed force-offs. The primary difference in these modes is in the manner the excess time from one non-coordinated phase is used by another non-coordinated phase. The non-coordinated phases can gap out if they have detectors and are operated in an actuated manner. A force-off point for each non-coordinated phase is the point in the cycle where the respective phase must terminate to ensure that the controller returns to the coordinated phase at the proper time in the cycle[76].

- *Floating force-off* : In floating force-off mode, the duration of the non-coordinated phases is limited to the splits that were programmed in the controller. As a consequence floating force-off does not allow for any time from phases with excess capacity to be used by a phase with excess demand. This means that phases that are allowed to start earlier as a consequence of an excess of capacity on phases earlier in the cycle will be forced to terminate before their force-off point in the cycle. This results in an early return to the coordinated phases. Let us take an example of a four phase signal. If the green required for the phase 1 is much lower than the vehicle inflow capacity, the green time of phase 1 is terminated after the minimum force-off period and the remaining green time available is not available to other phases. The second phase starts earlier than the starting period and continues recursively for all the phases. The second cycle begins much earlier if there is some amount of green time greater than the maximum cycle length allowable.

- *Fixed force-off* : Fixed force-off, on the other hand, allows the transfer of excess capacity from one phase to a subsequent phase with excess demand. This means that phases with excess demand will terminate at the force-off point irrespective of when the phase starts. The controller only allows the use of excess unused capacity and ensures that coordinated operations are not disrupted.

Some of the advantages and disadvantages of fixed force-off are :

- Fixed force-off allows better utilization of the time available from phases operating below capacity by phases having excess demand that varies in a cyclic manner. This is the case when phases earlier in the phasing sequence operate below capacity more often than phases later in the phasing sequence.

- Fixed force-off minimizes the early return to coordinated phases, which can be helpful in a network that has closely spaced intersections. An early return to the coordinated phase at a signal can cause the platoon to start early and reach the downstream signal before the onset of the coordinated phase, resulting in poor vehicle progression.

- Fixed force-off minimizes the early return to the coordinated phase, which is a major disadvantage. Under congested conditions on arterial roads, an early return can result in the queue clearance for coordinated phases. Minimizing early return to coordinated phases can cause significant disruption to coordinated operations and in the dispersion of the platoons. This disadvantage can be overcome by adjusting the splits and/or offsets at the intersection to minimize disruption.

Overall, fixed force-off has the potential to improve signal operations by better utilization of any excess capacity. However, fixed force-off will only benefit if the phases that are more likely to be below capacity are earlier in the phasing sequence and allows the maximum utilization of the maximum available green time.

3.1.3 Traffic adaptive control

Traffic load is highly dependent on parameters such as time, day, season, weather, and unpredictable situations such as accidents, special events, or construction activities. These factors are taken into account by a traffic-adaptive control system, so that bottlenecks and delays can be prevented. Adaptive traffic control systems continuously sense and monitor traffic conditions and adjust the timing of traffic signals accordingly. Adaptive systems, like SCOOT (Split, Cycle and Offset Optimization Technique) and SCATS (Sydney Coordinated Adaptive Traffic System), have been around since the mid 70's, and have proven their worth in various places around the world. Using real-time traffic information, an adaptive system can continuously update signal timings to fit the current traffic demand. The aging of traffic signal plans, with a gradual degradation of performance as traffic patterns drift away from those in place during implementation, is well documented [77]. Many agencies have no program for monitoring the applicability of signal timing to the current traffic patterns, and it is not uncommon to find agencies that have not re-timed their signals in years. The benefits of an adaptive signal control system are apparent, since both traffic operations and staff can be made more efficient since a better performance can be gained with the same level of effort [78].

Adaptive traffic control systems are often categorized according to their generation. First-generation traffic-adaptive systems employ a library of pre-stored signal control plans, which are developed off-line on the basis of historical traffic data. Plans are selected on the basis of the time of day and the day of the week, directly by the operator, or by matching from an existing library a plan best suitable for recently measured traffic conditions. First generation traffic-adaptive systems are often referred to as traffic-responsive signal control. A limitation of traffic-responsive signal control is that by the time the system responds, the registered traffic conditions that triggered the response may have become obsolete or could have varied significantly. Second-generation traffic-adaptive systems uses an on-line strategy that implements signal timing plans based on real-time surveillance data and predicted values. The optimization process can be repeated every five to fifteen minutes . However, to avoid transition disturbances, new timing plans cannot be implemented more than once every 10 minutes.

Third generation traffic-adaptive systems are similar to the second-generation systems, but differ with respect to the frequency with which the signal timing plans are revised. The third generation of control allows the parameters of the signal plans to change continuously in response to real-time measurement of traffic variables, which allows for “acyclic” operation. Generally speaking, given time-varying unpredictable demand patterns, a traffic-adaptive system should be able to outperform a fixed time or actuated system. The margin of improvement demonstrated by a traffic-adaptive system over a fixed-time or traffic-actuated system cannot be compared easily to that determined for another adaptive system as it is strongly related to the network geometry and traffic demand chosen in the benchmark study. For a fair comparison, the systems should be benchmarked using the same test environment and an equal amount of effort should be put in the optimization of the different systems by people that are knowledgeable. The systems described in this section are systems from the, proven second generation (SCATS/GLIDE, SCOOT, MOTION) and from the, younger, third generation (OPAC, PRODYN, RHODES, UTOPIA/SPOT, TUC, HMS).

3.1.3a SCATS/GLIDE

SCATS (Sydney Coordinated Adaptive Traffic System) [79] was developed in the early 1970’s by the Roads and Traffic Authority of New South Wales, Australia. The system utilizes a distributed, three-level, hierarchical system employing a central computer, regional computers, and local intelligent controllers to perform a large-scale network control. The regional computer can execute adaptive control strategies without any aid from the central computer, which only monitors the system performance and equipment status. The control structure enables SCATS to expand easily and suitably for controlling any size of traffic area.

SCATS employs a strategic optimization algorithm and a tactical control technique to perform system-wide optimization. The optimization philosophy contains four major modules:

- Cycle length optimizer
- Split optimizer
- Internal offset optimizer
- linking offset optimizer

SCATS selects combinations of *cycle*, *splits* and *offset* from predetermined sets of parameters with few on-line calculations. Maximum freedom consistent with good coordination is given to local controllers to act in the traffic-actuated mode. The system is designed to automatically calibrate itself on the basis of data received, minimizing the need for manual calibration and adjustment. For control purposes, the total system is divided into a large number of comparatively small subsystems varying from one to ten intersections. As far as possible, the subsystems are chosen so that they can be run independently for many traffic conditions. For each subsystem, minimum, maximum, and geometrically optimum cycle lengths are specified.

To coordinate larger groups of signals, subsystems can link together to form larger systems, operating on a common cycle length. Linking plans manage the linking between subsystems. When a number of subsystems are linked together, the cycle time becomes a linked subsystem with the longest cycle time. The combination of subsystem plans, link plans between subsystems, variable cycle length, and variation of offsets provides an infinite number of operating plans. Four background plans are also stored in the database for each subsystem. The cycle length and the appropriate plan are selected independently of each other to meet the traffic demand. For this purpose, a number of detectors in the subsystem area are defined as strategic detectors; these are stop-line detectors at key intersections. Various system factors are calculated from the strategic detector data, which are used to decide whether the current cycle and plan should remain the same or be changed. Strategic options, minimum delay, minimum stops, or maximum throughput can be selected for the operation. These options can be permanent or dynamically changed at threshold levels of traffic activity. Four modes of operation are included in SCATS:

- *Masterlink Operation* : This is the normal mode of operation which provides integrated traffic-responsive operation. There are two levels of control in this mode: strategic and tactical. Strategic control determines the best signal timings for the areas and subareas, based on average prevailing traffic conditions. Tactical control is concerned with the control of the individual intersections within the constraints imposed by the strategic control. This lower-level control deals with termination of the unnecessary green phases when the demand is below the average. The basic traffic measurement used by SCATS for strategic control is a measure analogous to the degree of saturation on each approach. This measure is used to determine cycle length, splits, and the direction and magnitude of offset.

- *Flexilink Operation* : In the event of failure of a regional computer or loss of communications, the local controllers can revert to a form of time-based coordination. In this mode, adjacent signals are synchronized by reference to the power mains frequency or an accurate clock, and signal timing plans are selected by time-of-day. The local controller operates under a vehicle-actuated or a fixed-time control system.

- *Isolated Operation* : In this mode, the controller operates under independent vehicle actuation or a fixed-time control system.

- *Flash Operation* : This is a manual mode in which normal automatic operation is overridden. It incorporates flashing yellow display for the major approaches and flashing red display for the minor approaches.

SCATS has also been widely used in several cities in Australia, New Zealand, USA, China, Singapore, Philippines, and Ireland. GLIDE (Green Link Determining) is a version of SCATS adapted according to the traffic network structure and requirements in Singapore.

3.1.3b SCOOT

SCOOT (Split, Cycle, and Offset Optimization Technique) [80, 81] was initiated by the British Transport and Road Research Laboratory (TRRL) in the 1970’s, with its first commercial system installed in 1980. SCOOT is a centralized system based on a traffic model with an optimization algorithm adapted for on-line application. Optimization takes place by incrementally updating a fixed-time plan. The benefit of this approach is that changes are gradual. The transition is less disruptive and less prone to overreacting than the transition between distinct plans as is typical in a time-of-day scheme. SCOOT performs optimization at three levels: Split, Cycle and Offset. SCOOT measures vehicles with a detector at the upstream location of the stop line.

SCOOT predicts the number of vehicles arriving at the intersection based on the updated flow information collected by the upstream detectors. Difference between the predicted arrival count and the actual departure count values gives the number of vehicles in queue. The predicted flow profile and traffic count are estimated for each cycle from a combination of the vehicles approaching, the time to clear the queue, the impact of offset and split adjustment. The split optimizer in SCOOT evaluates the projected arrival and departure profiles every second. A few seconds before the change of signals, the system adds the delay from all movements that will end or begin at that change of signals. This delay is compared with the delay calculated for the change of signals occurring either a few seconds earlier or later. The best balance of movement that provides the least delay is implemented.

The offset optimizer operates on each node pair and searches for the best offset timing to improve traffic progression on the basis of the cyclic profile. Based on the profile measured in the previous cycle, the offset optimizer minimizes the delay for all movements of the intersection by incrementing or decrementing the current offset with a few seconds. With the offset optimizer in the SCOOT systems, green waves can be imposed along the coordinated signal controlled corridor. After this offset adjustment, the split optimizer may further adjust the signal timings based on traffic actually approaching the stop line at that time.

The cycle optimizer looks at the saturation levels of all intersection movements once each cycle-control period. At critical intersections with low reserve capacity, the cycle optimizer will extend the length of the cycle. It does so in different increments of time (e.g., 4, 8, or 16 seconds) depending on the current cycle length. If an intersection is operating below capacity, the cycle optimizer will reduce the length of the cycle. SCOOT has been widely used in several cities in UK, USA, Canada, China, South Africa, Cyprus, Pakistan, United Arab Emirates, Chile, and Spain.

3.1.3c MOTION

MOTION (Method for Optimization of Traffic signals In Online-controlled Network) [82, 83] is a traffic signal control strategy developed by Siemens, Germany. The system operates on three functional levels: on the strategic level, every 5, 10 or 15 minutes (cycle time, average green time distribution, basic stage sequences and network coordination); on the tactical level, every 60 to 90 seconds (cycle, current stage sequence); and on the operational level, every second (green time modification). Starting with the dominant traffic stream through the network, a grid of green waves is constructed, taking into account modelled (or if available, measured) platoons in the links. For each intersection, the optimum sequence of stages is identified, and the basic split of green times is fixed.

Depending on the remaining spare time per intersection, and on the constraints of the optimized offsets, a certain amount of bandwidth is available for the subsequent local optimization. Optimization normally aims at minimizing delays and stops in the network. In the final step the decision is made to change the signal programs at the intersections. To avoid frequent minor changes, changes are only implemented if calculation determines a significant improvement in the overall optimization objective. Depending on the type of local controller and on the local control method used, the signal programs are then converted and implemented. To avoid severe disruptions in traffic flow due to the plan switch, a smooth (gliding) transition from the running to the new plan is performed. Until the next optimization run of the network model, the local controllers operate on their own and modify their plan according to the local situation, but always staying within the given bandwidth.

3.1.3d TUC

TUC (Traffic-responsive Urban Control) [84] employs a store-and-forward based approach to road traffic control, which introduces a model simplification that enables the mathematical description of the traffic flow process without the use of discrete variables. This opens the way to the application of a number of highly efficient optimization and control methods (such as linear programming, quadratic programming, nonlinear programming, and multi-variable regulators), which, in turn, allow for coordinated control of large-scale networks in real-time, even under saturated traffic conditions.

The critical simplification is introduced when modelling the outflow of a stream suggests that there is a continuous (uninterrupted) outflow from each network link (as long as there is sufficient demand). The consequences of this simplification are:

- The time step *t* of the discrete-time representation cannot be shorter than the cycle time *C*, hence real-time decisions cannot be taken more frequently than at every cycle.
- The oscillations of vehicle queues in the links due to green/red-commutations are not described by the model.

The effect of offset for consecutive intersections cannot be described by the model. Despite these consequences, the appropriate use of store-and-forward models may lead to efficient coordinated control strategies for large-scale networks. The three main modules of TUC are the split, cycle, and offset control modules that allow for real-time control of green times, cycle times and offset. The basic methodology employed for split control by TUC is the formulation of the urban traffic control problem as a Linear-Quadratic (LQ) optimal control problem based on a store-and-forward type of mathematical modelling.

The control objective is to minimize the risk of oversaturation and queue spill-back, and this is achieved through the appropriate manipulation of the green splits at signalized junctions for given cycle times and offsets. Longer cycle times typically increase the capacity of the junction as the proportion of the lost time caused by switching signals becomes accordingly smaller. Longer cycle times may however increase vehicle delays at under saturated junctions with longer waiting times during the red phase. The objective of cycle control is to increase the capacities of the junctions as much as necessary to limit the maximum observed saturation level in the network.

Within TUC this objective is achieved through the application of a simple feedback-algorithm that uses the maximum observed saturation levels of a pre-specified percentage of the network links as the criterion for increase or decrease of the cycle time. Offset control is achieved through the application of a decentralized feedback control law that modifies the offsets of the main stages of successive junctions along arterials, so as to create "green waves" when possible, taking into account the possible existence of vehicle queues. To implement a new offset in TUC, a transient cycle time is temporarily implemented at all but the first junction along an arterial road. The transient cycle time is implemented one single time, after which all the junctions along the arterial road are coordinated according to the new offset.

3.1.3e UTOPIA/SPOT

UTOPIA/SPOT (Urban Traffic Optimization by Integrated Automation/Signal Progression Optimization Technology) [85] is a traffic signal control strategy developed by Mizar Automazione in Turin, Italy. UTOPIA/SPOT calculates optimal control strategies for subareas of the network, with each subarea having the same cycle length. While operating, the system maintains a historical database of measured flows, turning percentages, saturation flows, and cycles in use.

The system utilizes a distributed, two-level, hierarchical system employing a central, area-level computer, and intersection, local-level computers to perform large-scale network control. SPOT is a fully distributed, traffic-adaptive signal control system, which operates by performing a minimization of local factors such as delays, stops, excess capacities of links, stops by public or special vehicles, and pedestrian waiting times. With each repetition, all SPOT units exchange information on the traffic state and preferential policies with their neighbouring SPOT units. This permits the application of look-ahead (each SPOT unit receives realistic arrival predictions from upstream intersections) and strong interaction (each controller considers, in the local optimization, the adverse effects that it could have on downstream intersections). Data is exchanged with neighbouring intersections every few seconds.

As each SPOT-unit communicates with surrounding units, the system can be programmed to prioritize public transport and emergency vehicles by giving early warning of these vehicles or by allowing them to be quickly cleared through the intersection. SPOT can also prioritize traffic on the basis of adherence to timetables, number of passengers, etc. SPOT allows a staged system implementation over time starting with a few intersections. It can be implemented without a central computer for small systems of typically six intersections or less. However, for larger intersection networks, the UTOPIA central PC-based control system should be added.

At the area level, the UTOPIA-module provides a mechanism to handle critical situations in the form of two actions that a signal controller may request of adjacent signal controllers. Thus, a controller may cope with congestion by requesting that a downstream signal increase throughput or that an upstream controller decrease demand. These requests are realized by respectively relaxing or tightening green time constraints . For the area level, UTOPIA, the model can:

- Analyze area-wide traffic data and make predictions for main street flows over time
- Apply its internal macroscopic model to entire area network and traffic counts

- Optimize the total travel time with constraints of average speed and saturation flows

SPOT has been used in several cities in Italy, The Netherlands, USA, Sweden, Norway, Finland, Denmark, and the UK.

3.1.3f OPAC

The OPAC (Optimized Policies for Adaptive Control) algorithm [86-88] has gone through several development cycles ranging from OPAC I through OPAC-VFC. OPAC maintains the specified phase order. For uncongested networks, OPAC uses a local level of control at the intersection to determine the phase on-line, and a network level of control for synchronization, which is provided either by fixed-time plans (obtained off-line), or a "virtual fixed cycle". A virtual fixed cycle is a cycle that although fixed between intersections (to enable synchronization), is determined on-line (hence virtual). Predictions are based on detectors located approximately 10-15 seconds upstream. After the initial 10-15 seconds, a model predicts traffic patterns (typically 60 seconds).

OPAC breaks the signal optimization problem into sub problems using dynamic programming, an approach that leads to a more efficient computation. At the same time it determines a virtual cycle. These are implemented for a time-step (roll period) of about 2-5 seconds. The length of the virtual cycle is varied according to the needs of either the critical intersection or the majority of intersections. The virtual cycle is allowed to change by typically one second per cycle. Within this limitation, OPAC provides local coordination by considering flows into and out of an intersection in selecting its offset and phase lengths. The congestion control process in OPAC generally attempts to maximize throughput, by selecting the phase that will allow the maximum number of vehicles to pass the intersection. OPAC does this by considering saturation flows and space available to store vehicles on each link. The first step of congestion control involves determining the next phase given that there is not a critical link that is on the verge of or currently experiencing spill-back. On the basis of these calculations, the algorithm determines whether it is necessary to revisit the timings at neighbouring intersections in light of throughput constraints that their physical queues impose on each other's effective service rates.

OPAC-I assumes an infinite horizon and uses dynamic programming to optimize the performance index. OPAC-I cannot be implemented on-line in real-time because of the extensive time required to compute the optimal settings. OPAC II used an optimal sequential constraint search (OSCO) to calculate the total delay for all possible phase switching options. The optimal solution was determined as the phase switching that produces the lowest total delay values, and OPAC-II was found to derive solutions with performance indexes within 10% of those generated with OPAC-I. Although OPAC-II was faster than OPAC-I, it still suffered from the need for vehicle arrival information for the entire planning stage, which was 50-100 seconds in length. OPAC-III was the first version of OPAC that featured the rolling horizon approach and was developed at first for a simple two-phase intersection, but later extended to an eight-phase intersection, which allowed phase skipping. OPACVFC added the algorithm used to coordinate adjacent signals.

3.1.3g PRODYN

PRODYN (Programmation Dynamique) [89, 90] is a real-time traffic control algorithm, which has been developed by the Centre d'Etudes et de Recherches de Toulouse (CERT), France. PRODYN evolves from two stages of development: two-level hierarchical control (PRODYN-H) and then decentralized control (PRODYN-D). The former offers the best result; however, its applicability is restricted due to the complex computations involved and the network size (limited to about 10 intersections). The latter, on the other hand, alleviates those limitations. Two approaches have been studied for PRODYN-D: no exchange (PRODYN-D1) versus exchange (PRODYN-D2) of information between the intersections. At the intersection level, the optimization model's aim is to minimize delay by using improved forward dynamic programming with constraints on maximum and minimum greens.

At the network level, the network coordination optimization is performed by a decentralized control structure. The procedure includes:

- Simulating a specific intersection output for each time step as soon as the intersection controller finishes its optimization over the time horizon
- Sending the simulation output to each downstream intersection controller
- Using the output message from upstream controllers at the next time step to forecast arrivals.

3.1.3h RHODES

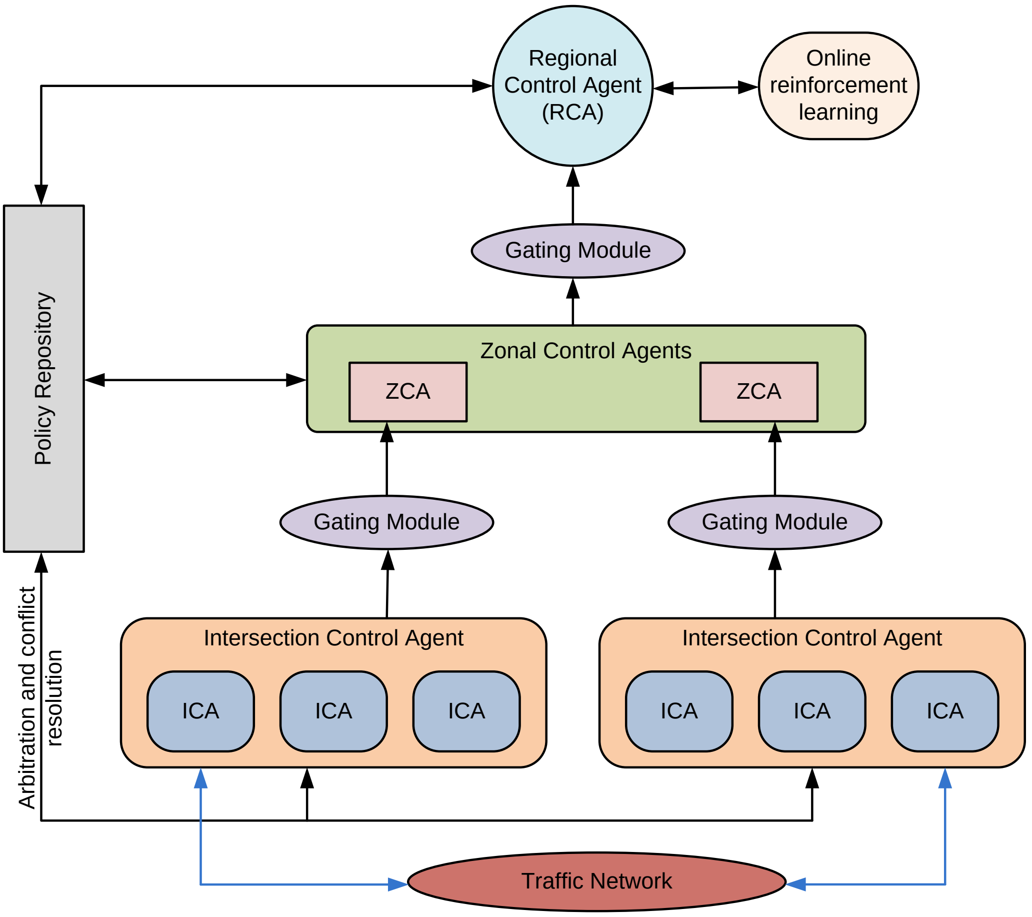
RHODES (Real-Time, Hierarchical, Optimized, Distributed, and Effective System) [91] is a hierarchical control system that uses predictive optimization, allowing intersection and network levels of control. RHODES includes a main controller, a platoon simulator (APRES-NET [92]), a section optimizer (REALBAND [93]), an individual vehicle simulator (PREDICT [92]), and a local optimizer (COP [94]). RHODES requires upstream detectors for each approach to the intersections in the network. RHODES also can use stop-line detectors to calibrate saturation flow rates and to improve traffic queue estimates. RHODES is entirely based on dynamic programming, and it formulates a strategy that makes phase switching decisions based on vehicle arrival data. The design of RHODES is based on dividing the traffic control problem into sub problems by use of a network hierarchy. The sub problems include the network-loading problem, the network flow control problem, and the intersection control problem.

At the top of the hierarchy is the network-loading problem. At this level, link loads and the prediction of the trends in the change of loads from real-time data are estimated. RHODES uses this information pro-actively to predict future platoon sizes near the boundaries of the system. The middle level consists of the network flow problem and involves the selection of signal timing to optimize the overall flow of vehicles in the network. The decisions are made in this level every 200-300 seconds. A platoon prediction logic model called REALBAND is used at this level. Network optimization is also established at this level and its results are used as constraints for the decision made in the next level. The lowest level of the control strategy is the one at the intersection and it is responsible for making the final second-by-second decisions regarding traffic signal operation. This level uses two sublevels of logic.

The first is the Link Flow Prediction Logic which uses data from detectors on the approach of each upstream intersection, together with information on the traffic state and planned phase timings for the upstream intersection, to estimate vehicle arrivals at the intersection being optimized. The other level is the Controlled Optimization of Phases (COP), which uses the information from the network flow problem, in addition to the results from the link prediction logic, to determine whether the current phase should be extended or terminated.

3.1.3i Hierarchical multi-agent system (HMS)

The hierarchical multi-agent signal control model is designed in an hierarchical manner to provide different levels of control for the traffic network. The architecture consists of three layers with each layer having a different internal structure and composition. The agent in the lowest layer consists of intersection controller agents (ICA) that control individual, pre-assigned intersections in the traffic network. The middle layer consists of zone controller agents (ZCA) that controls several pre assigned ICAs. The highest level consists of one regional controller agent (RCA) controls all the ZCAs.



Regional Control Agent (RCA)
Online reinforcement learning
Gating Module
Policy Repository
Zonal Control Agents
ZCA
ZCA
Gating Module
Gating Module
Intersection Control Agent
ICA
ICA
ICA
Intersection Control Agent
ICA
ICA
ICA
Arbitration and conflict resolution
Traffic Network

Figure 3.1. Architecture of hierarchical multi agent system

The three-layered multi-agent architecture [45] is shown in Figure 3.1. The problem of real-time network-wide signal control is divided into several sub-problems of different scales and magnitude. Individual agents from each layer of the multi-agent architecture are tasked to manage the respective sub-problems according to their position in the hierarchy. Each agent is a concurrent logical process capable of querying, directly interacting with the environment (e.g., sensors and agents in the lower hierarchy within its control) and making decisions autonomously. The agents in each layer decide the appropriate local policies that indicates the green signal timings necessary for each phase and levels of cooperation- indicated by cooperative factor that they deem appropriate based on the conditions of the intersections, or set of intersections under their jurisdictions. Besides having higher-level traffic network parameters as inputs to their decision-making process, the higher-level agents obtain the cooperative factors recommended by their lower level agents as inputs too (Fig.3.1. shows that the intersection cooperative factors recommended by the lower level ICAs are part of the inputs of a ZCA).

Based on these inputs, the decision-making process of the higher level agents may present a set of higher-level policies that are different from those policies recommended by their lower level agents or they may choose to follow the lower level policies. The policy repository is a dynamic database for storing all the policies recommended by the controller agents of all levels at the end of each evaluation period. The end of an evaluation period is indicated when all the intersections have finished their current signal phases. After each period, the previously recommended policies are updated with a new set of policies. The policy repository then performs arbitration and conflict resolution for the entire set of recommended polices.

The arbitration process gives priority to higher-level policies. However, since one of the outputs, namely the cooperative factor, of the lower-level agents is a part of the inputs to higher-level agents (as mentioned earlier), the lower-level decisions affect directly the outcomes of the higher-level agent’s decision-making process. As such, lower-level policies are not completely neglected by the arbitration process. The decided final policies for each of the agents in different hierarchy is stored in the policy interpreter for future use. The function of the policy interpreter is to translate the chosen set of policies into actions, which may result in adjustment of the various signal-timing parameters such as phase-length, cycle-time, direction of offset coordination, for the affected intersections.

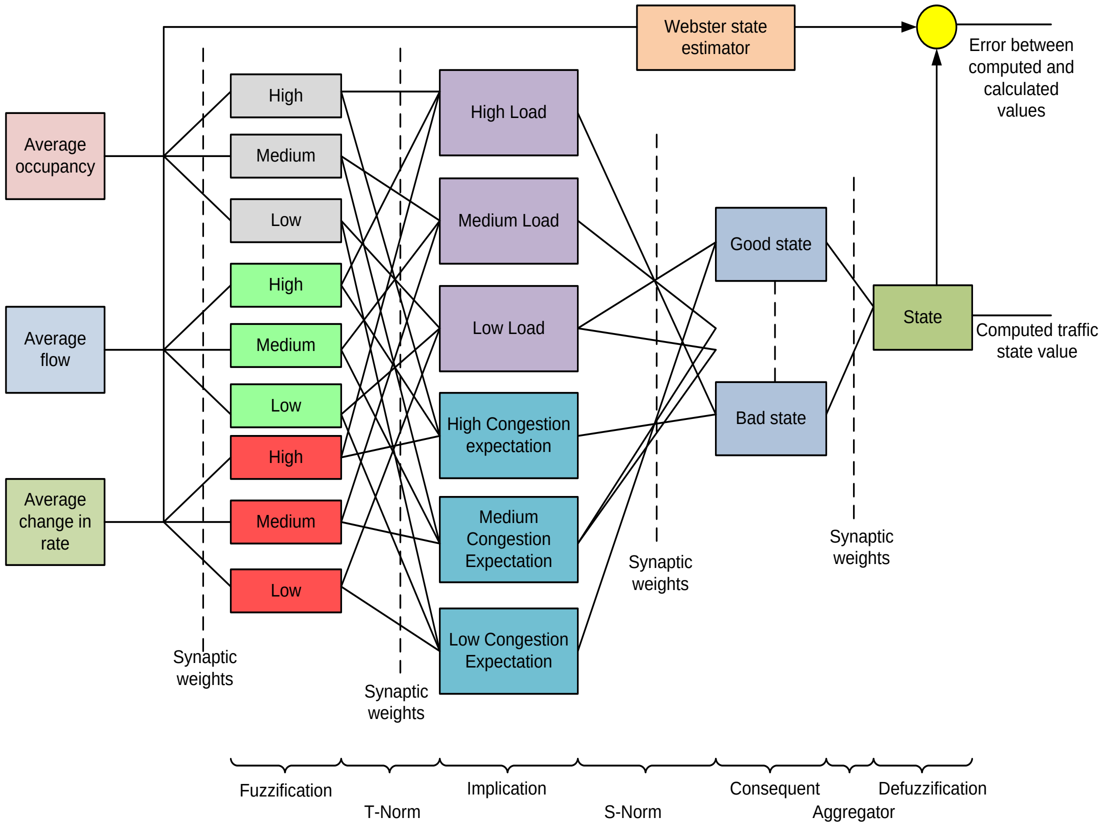


Figure 3.2. Internal neuro-fuzzy architecture of the decision module in zonal control agent

Each Intersection Control Agent (ICA) takes in the lane-specific occupancy, flow and rate of change of flow of the different intersection approaches as input. The occupancy, flow and rate of change of flow are computed from the data collected by loop detectors placed at the stop line of the intersection in the lanes with right of way during the specific phase in progress. Based on the collected data, green time for a specific phase and the intersection cooperation factor is computed. In order to quantify the traffic conditions of the intersections in a zone, neuro-fuzzy decision module of the ZCA takes in each intersection's representative occupancy, flow and rate of change of flow as its inputs is shown in Figure 3.2.

The fuzzy sets of occupancy, flow and rate of change of traffic volume have three linguistic labels, namely high, medium and low to describe the respective degrees of membership (Gaussian membership function). Besides these inputs, the ZCA also takes in the intersection cooperative factors recommended by the respective ICAs under its control (to reflect the level of cooperation each ICA sees fit for its own intersection, all of which are within the zone controlled by the ZCA). The antecedents of the fuzzy rules are defined by properly linking the nodes in the second layer to those in the third layer. The third layer fires each rule based on the T-norm fuzzy operation, implemented using the minimum operator. Nodes in the third layer define the degree of current traffic loading of the zone (i.e., high, medium and low loads) and the level of cooperation needed for the intersections within the zone (i.e., high, medium and low degrees of cooperation). The nodes in the fourth layer represent the various consequents that correspond to the fuzzy rules in the decision module.

For the signal policy inference engine, the consequents consist of the various signal improvement/control policies. For the cooperation factor inference engine, the consequents consist of the various possible levels of cooperation. Since some of the fuzzy rules share the same consequent, the S-norm fuzzy operation is used to integrate and reduce the number of rules. For this research, the S-norm fuzzy operation is implemented using the MAX operator.

Finally, the fifth layer performs the defuzzification process in order to obtain crisp values correspond to the chosen signal policy and cooperative factor (i.e., outputs of the neuro-fuzzy decision module for each agent). The architecture of the decision module for the ICA and RCA is largely similar to the one described for the ZCA. The main difference lies in the inputs and the hierarchical nature of the overall multi-agent architecture. The RCA can overrule the decisions of all agents in the lower level of hierarchy than itself.

The flow of control is from top to down and no sideways communication between agents of same hierarchy exists. This effectively reduces the amount of information to be communicated between agents. This reduces the effects of information corruption, noise addition or the loss of data arising due to communication link failure. However, data mining requirements are extremely inflated. It is not an easy task to identify the representative input data required by agents in higher level of hierarchy. The complexity further increases with growing network size.

3.2 SUMMARY

This chapter presented a comprehensive review of the existing traffic signal control techniques and highlighted the advantages and disadvantages of the various traffic-adaptive control strategies. The review clearly indicates the lack of a distributed intelligent traffic signal control capable of optimizing the green time minimizing the total travel time delay experienced by the vehicles inside the road network. The insights that are derived from the previous research works will be useful in designing a multi-agent based distributed traffic signal control with effective communication and learning capabilities.

CHAPTER 4

DESIGN OF PROPOSED MULTI-AGENT ARCHITECTURE

In this chapter, the proposed multi-agent architecture is presented in detail. Based on the detailed study of the advantages and disadvantages of the various agent architecture presented in Chapter 2, a connected distributed architecture with local communication through message passing between agents is proposed. For proper functioning of any multi-agent system, it must be designed in a modular fashion with each module performing a specific task. The basic functional aspects that need to be present in a multi-agent architecture are identified, and the required interactions with different modules within the agent are explained in detail.

4.1 PROPOSED AGENT ARCHITECTURE

The proposed multi-agent traffic signal control system is a distributed system with autonomous agents capable of interacting with each other and deciding the optimal action without an external supervisory system. The overall structure of the proposed multi-agent architecture is shown in Figure 4.1. It consists of nine distributed agents communicating with their immediate neighbours to perform the desired action.

The proposed agent system does not have a specific overall agent management structure except for the directory service as all the agents have homogenous structure and equal decision making capabilities. The directory service stores each agent's attributes, behaviours, information pertaining to the neighbouring intersection, number of links, number of lanes etc. The structure of each individual agent is shown in Figure 4.2. In order to control the traffic flow at urban road network, each intersection is controlled by an autonomous agent.

Each intersection agent is composed of six modules or functional blocks operating concurrently. They are as follows:

- Data Collection module
- Communication module
- Decision module
- Knowledge base and data repository module
- Action implementation module
- Backup module

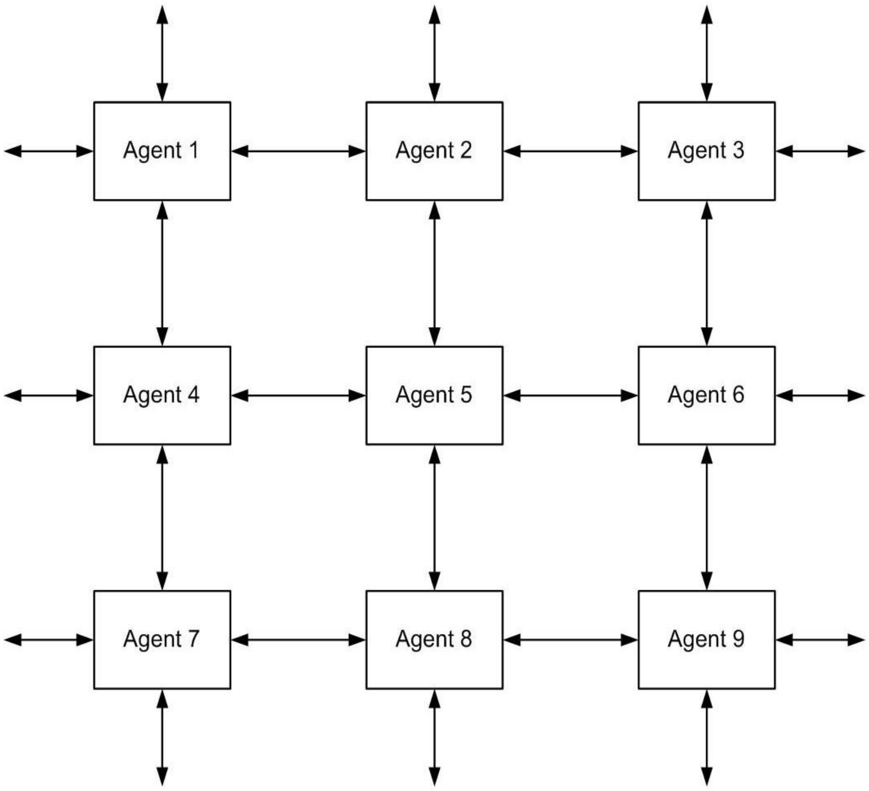


Figure 4.1. Overall structure of the proposed multi agent architecture

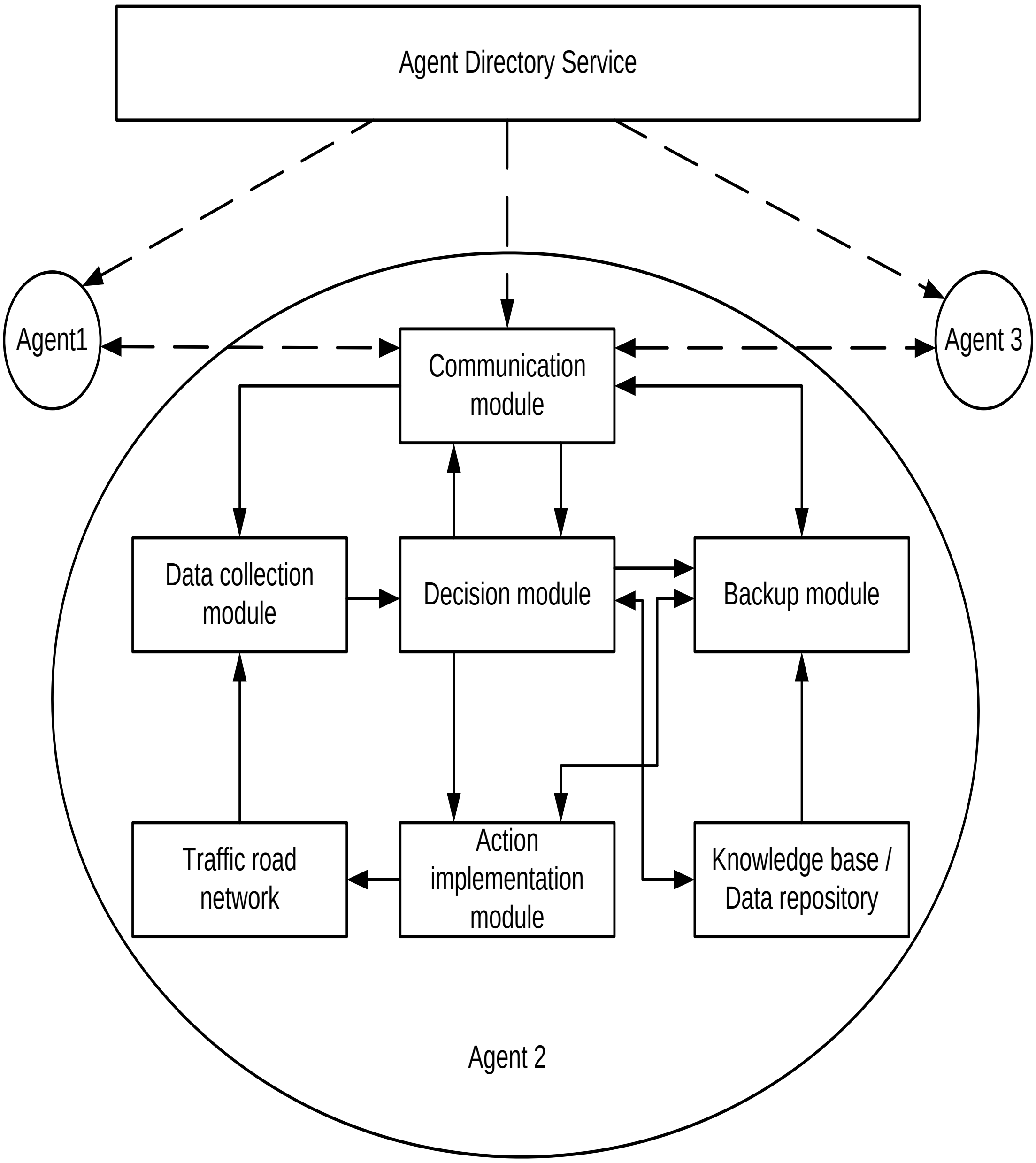


Figure 4.2. Internal structure of the proposed multi agent system

Each module performs a specific task and stores the output in a buffered memory thereby creating a concurrent or parallel environment. The architecture and the decision system of the individual modules can vary in accordance to the requirements of the application. Functionality of each of the module is described in detail in the following sections.

4.2 DATA COLLECTION MODULE

Each agent receives the data required for its functioning from the loop induction detectors placed near the stop line of the intersection. A snapshot of the intersection showing the induction loop detector is shown in Figure 4.3.

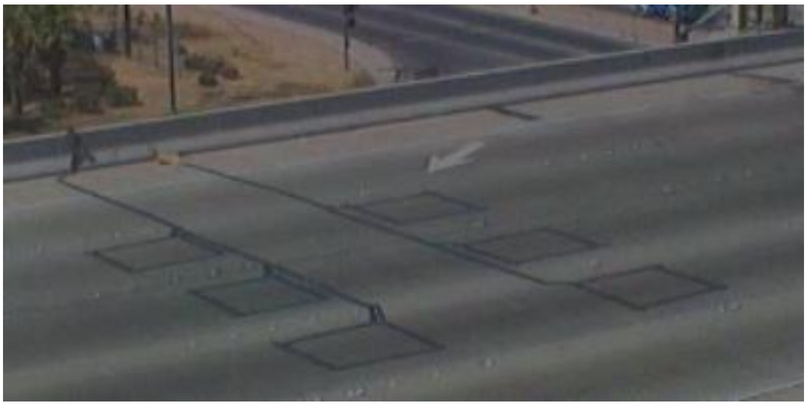


Figure 4.3 Induction loop detectors at intersection

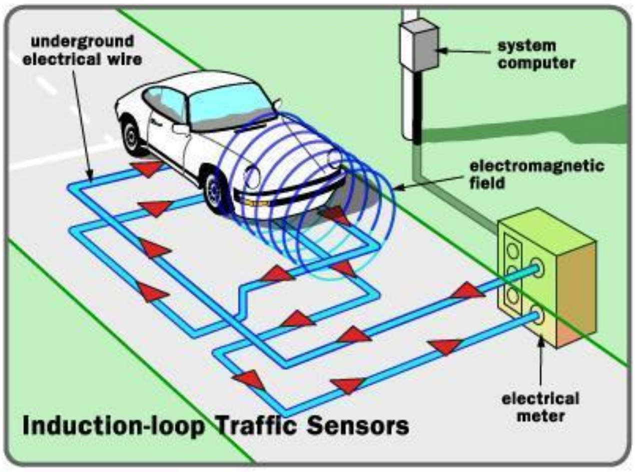


Figure 4.4. Working of Induction loop detectors

The induction loop detector placed on road detects the presence of a vehicle using the change in the magnetic field caused by the metallic parts of the vehicle. Using this information, the road side electronic devices can compute the number of vehicles that have crossed the intersection, the amount of time a vehicle occupies the link etc. Data collection module interacts with these electronic devices to gather the traffic information. The traffic data used by the agent system designed are flow, queue, rate of change of flow, and the ratio of green time at neighbouring intersections to maximum permissible green time. Not all of the input data is used in the designed agent system and the choice of input depends on the decision system employed.

The flow rate of vehicles in a link is calculated based on the difference between the upstream arrival flow rate and downstream departure flow rate for each lane in the link. Upstream refers to the intersection that releases the vehicles and downstream is the intersection that receives the vehicles. The data is sampled at the frequency of five to ten seconds.

Increasing the sampling frequency would enable collection of large amount of data that can be used to construct a realistic traffic arrival distribution and increase the possibility of noise compensation by averaging. Mathematically, the flow value calculated can be expressed as

$$f_{ph} = \frac{\sum_{k=1}^{c_g/\nabla T} \sum_{i=1}^{n} \sum_{j=1}^{l} \left(f_{ij}^{up} - f_{ij}^{down}\right)}{n.l.\left({}^{c_g}/_{\nabla T}\right)} \qquad (4.1)$$

where $c_g$is the current green time for the phase, n is the number of links having right of way during the phase, l is the number of lanes in the link, ∇T is the sampling period and,$f_{ij}^{up}, f_{ij}^{down}$ are the upstream and downstream vehicle flow rate respectively.

The queue information is calculated as the number of vehicles waiting in lanes which had the right of way during the phase that just ended. The queue value gives only the remaining vehicles count and does not take into consideration the vehicle count added after the end of a phase. As it is essential to calculate the green time based on the maximum congested lane in a link, the maximum value of queue formed is used as the input data. Equation (4.2) shows the mathematical expression for obtaining the maximum queue from the links.

$$Q = max\{(q_{11}^{up} - q_{11}^{down}) + q_{11}^{prev}, (q_{12}^{up} - q_{12}^{down}) + q_{12}^{prev}, \dots \dots \dots ., (q_{21}^{up} - q_{21}^{down}) + q_{21}^{prev}, \dots \dots \dots \dots\} \qquad (4.2)$$

where Q is the maximum queue value at an intersection, $q_{ij}^{up}, q_{ij}^{down}, q_{ij}^{prev}$ represents the upstream, downstream traffic count and the previous stored queue for the specific lane in a link respectively.

It is assumed that the arrival rate of vehicles can be approximated to a uniform distribution, as the average value is calculated from data sampled at high frequency, and can adequately compensate queue increase at the end of each phase.

The rate of change of flow is the other input that is used in few of the proposed agent architecture. Rate of change is computed as the difference between the average flow value computed at decision time instance t for a specific link and lane to the flow used during decision time $t - 1$.

$$rf(t) = \left\{\sum_{i=1}^{l} \sum_{j=1}^{la} \left(f_{ij}(t) - f_{ij}(t-1)\right)\right\} / l * la \qquad (4.3)$$

Where $rf(t)$ is the rate of change in flow at decision instance t, l is the number of links, la is the number of lanes connected to each link,$f_{ij}(t)$ and $f_{ij}(t-1)$ are the flow values computed at decision time instance t and $t-1$. The rate of change in flow gives an indication of the variation in vehicle count value and helps in differentiating a saturated network from unsaturated one. The sign of the rate of change in flow also gives an indication whether the vehicle count has incremented or decremented during the evaluation period from the previous one.

The final input used is the information communicated by the neighbouring intersection. Its value is computed as ratio of the green time for a phase in progress at the neighbouring intersection to the maximum permissible green time. Usually the maximum permissible green time is fixed at forty seconds as given in highway capacity manual(HCM).

4.3 COMMUNICATION MODULE

The communication module is responsible for multiple functions. The communication module interacts with the agent directory service to collect the information regarding the agents that are connected to the current agent through incoming and outgoing links and their corresponding configuration. The communication module receives the information of neighbour status data or the reward value computed by the neighbouring intersection. The communication module follows some of the guidelines of FIPA protocol[60] (Foundation for Intelligent Physical Agents). The communication is performed as a broadcast. Each agent checks for the agent ID in the transmitted information and if it matches the ID of the connected agents (Queried from directory service), the information is stored in a buffer with a time stamp for use at a later time.

The communication module also communicates the state of the traffic of the intersection as neighbourhood status data or reward value to the adjacent agents as a broadcast.

The module is responsible for coordination and cooperation between the agents. In absence of communication between agents, the limited sensing capability of the sensors associated with each agent provides only a partial view of the environment. This can cause selection of actions that can produce higher delays in long run because of the correlated agent environment. The communication module serves to enhance the view of each agent by sharing information with other agents to improve the action policy selected to achieve global optimal solution.

The communication module works in an asynchronous manner therefore necessitating the use of buffer memory to store the data that can be used for synchronization of information in the later stages in the decision module. A typical communication between the agents is shown in Figure 4.6. The dotted arrows indicate the data received by the agent from the agents connected to the incoming links and the solid arrows indicate data transmitted to the other agents. Apart from the above mentioned functionalities, the communication module also interacts with the directory service to receive the information that are essential for computing the green time. This is helpful in informing the agents any variations required in the signal plans, phases or reduction in the capacity of infrastructure.

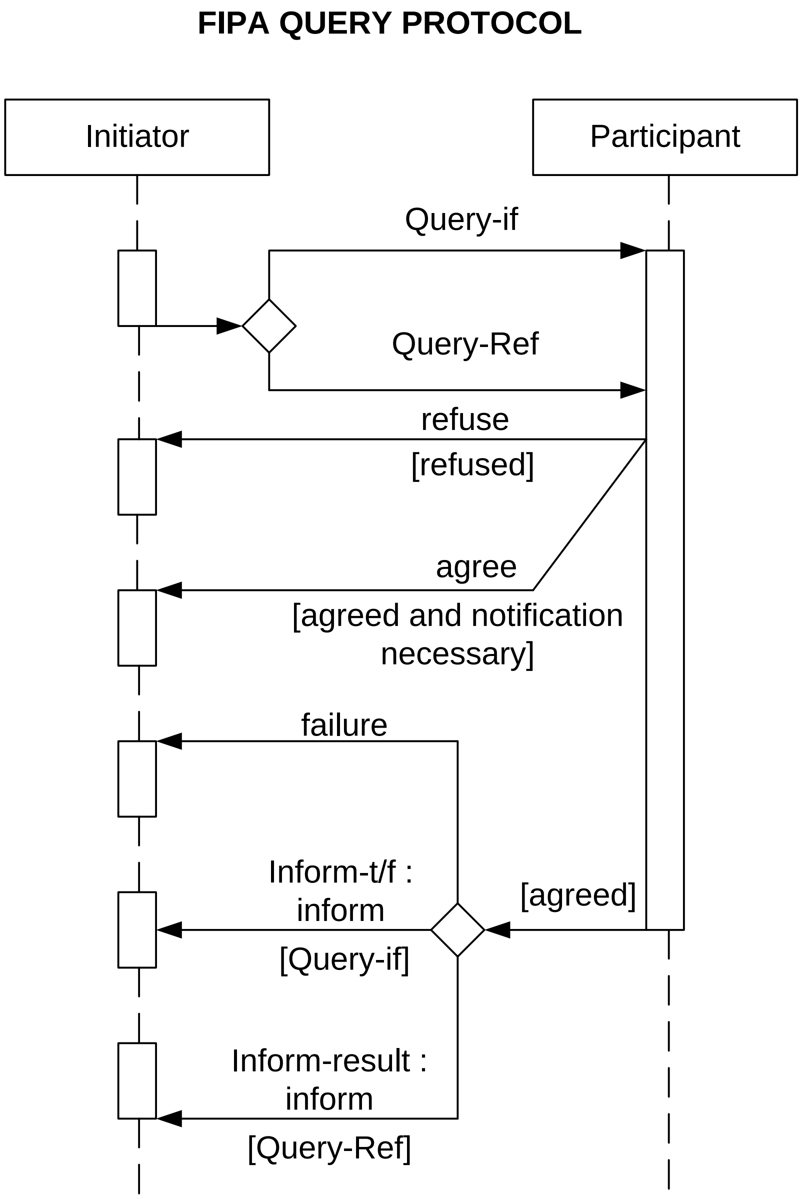


Figure 4.5. FIPA Query protocol[60]

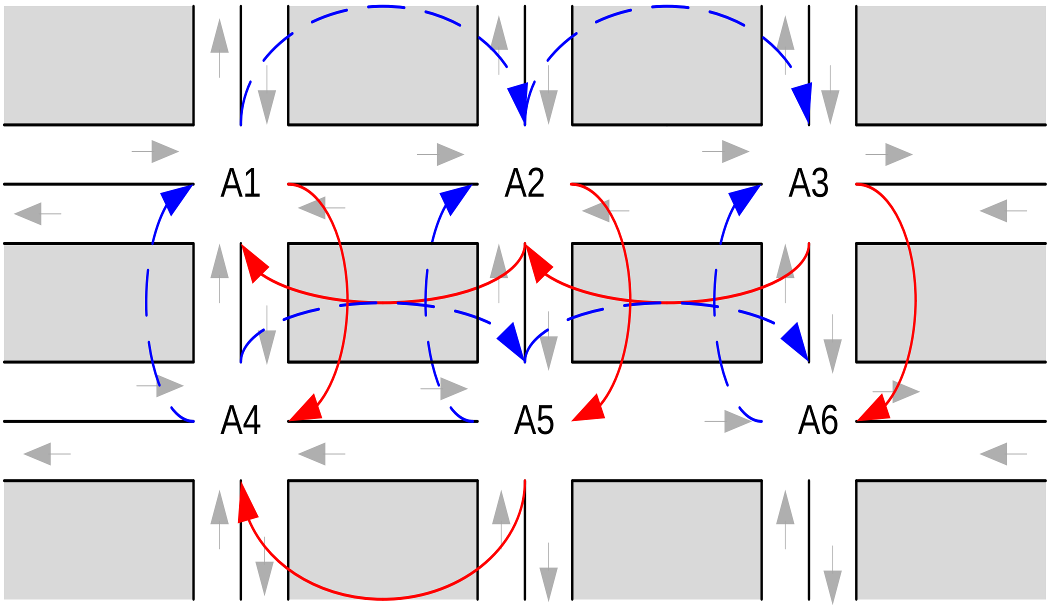


Figure 4.6. Typical communication flow between agents at traffic intersection

4.4 DECISION MODULE

The decision module is responsible for computing the optimal value of green time for each phase of signal at the intersection. The module uses the information received from the data collection module and the communication module to perform its action. The decision module could be designed using type-2 fuzzy or neuro-type-2 fuzzy system. The details of the proposed decision architecture are provided in Chapter 5.

4.5 KNOWLEDGE BASE AND DATA REPOSITORY MODULE

The knowledge base and data repository module is used to store the rule base and the data collected from the road networks to design fuzzy decision system and to perform online training of the designed decision system. The Knowledge base update is performed if there is any variation in the rule base or at an interval of 30 minutes, which is fixed to gather the data.

The knowledge update is performed in an asynchronous manner. Irrespective of whether the decision module is currently executing the decisions chosen or performing the computation operation, knowledge update is performed at pre-specified intervals. The knowledge base is also responsible for storing the signal plan calculated using the Webster's formula based on the historical traffic flow pattern. in case of failure to receive data from the road networks, these signal plans are used by the backup module.

4.6 ACTION IMPLEMENTATION MODULE

The action implementation module adjusts the green time of the phase in each cycle of a traffic signal control at an intersection based on the optimized green time computed by the decision making module. The action implementation module ensures the completion of all phases in a cycle before adjusting the green time so that vehicles in lanes that do not have right of way during a specific phase do not experience delay because of extension of the green time. In case of failure of the decision module due to certain reasons, the module communicates with the backup module to fetch the signal plan according to the timing of the day. In short, the action implementation module serves as a main communication interface between the traffic signal and the agent. It also allows for a central control to override the signal timings in case of emergency situations allowing manual control of the signals.

4.7 BACKUP MODULE

The backup module sets the signal control operation in pre-timed mode in the event of input failure for ten consecutive cycles or stationary action policy for twenty consecutive cycles. The backup module queries the knowledge base and data repository module for the appropriate signal plans based on the timing of the day. The signal plans stored in the knowledge base are computed based on Webster's formula. Procedure to obtain the lost time and calculation of split timing used in the Webster's signal time calculation is detailed in [29].

The backup system returns control to the agent once changes in the decision value are detected. This ensures smooth flow of traffic to the maximum possible extent except during network saturation or full capacity flow period, without affecting the network.

4.8 SUMMARY

The chapter covered the basic building blocks used to construct the proposed multi-agent architecture. It also provided a clear overview of the functionality of the individual blocks, the flow of control and data communication between modules and the necessity of modules to cooperate with other functional blocks to control traffic in an urban road network.

CHAPTER 5

DESIGN OF HYBRID INTELLIGENT DECISION SYSTEMS

In this chapter, details of the proposed decision systems design, architecture and their construction are discussed in detail. This chapter also presents an overview of type-2 fuzzy sets, symbiotic evolutionary learning techniques used in this thesis, their advantages and its application to urban traffic signal timing optimization problem.

5.1. OVERVIEW OF TYPE-2 FUZZY SETS

Fuzzy set theory or fuzzy logic was first proposed by Zadeh in early 1975 [95-97]. It was an attempt to represent the uncertainties and vagueness associated with the linguistic expressions both quantitatively and qualitatively. The transition from crisp set theory to fuzzy theory was to accommodate and exploit the generality of fuzzy theory and its ability to replicate the real-world scenario to a large extent. When there is no fuzziness involved in the definition of a particular class or cluster of objects, it becomes a simple two-valued characteristic function. The input or object is assigned a crisp value between zero and one if it belongs to the class and zero otherwise. This is not true in real-world applications. There is a certain level of abstraction and fuzziness associated with the membership values assigned to each input. Fuzzy logic is able to provide the abstraction similar to human thinking and account for this fuzziness.

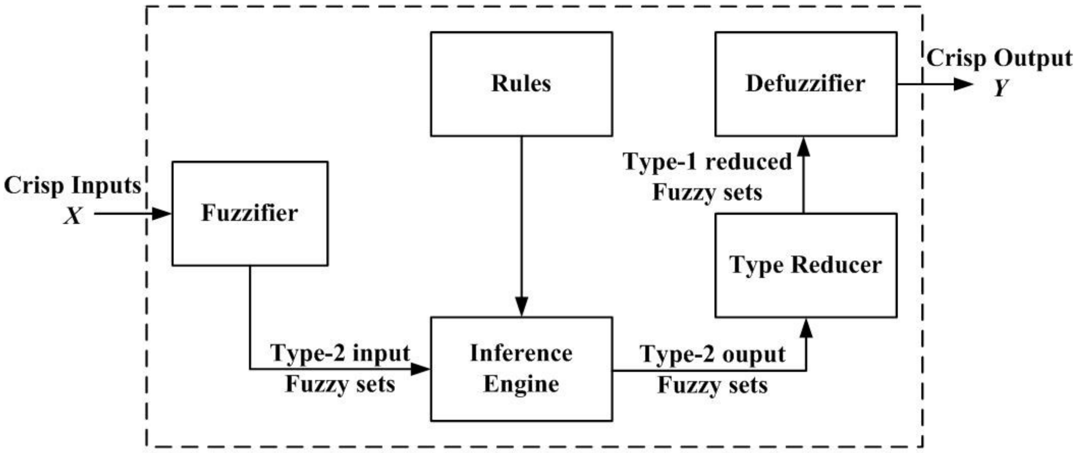


Figure 5.1. Block diagram of type-2 fuzzy sets

The preliminary fuzzy systems or type-1 fuzzy systems developed were based on the assumption of the availability of a crisp value of membership grade for the entire universe of discourse of the input. In many applications, determining a crisp membership value is a difficult task as an input could be perceived differently by individuals. For example, a traffic *flow* value of 250veh/hr could be assigned a value of 0.7 under class low and the same could be assigned a value of 0.4 by another individual. In order to handle such uncertainties in the interpretation, type-2 fuzzy sets were introduced. Type-2 fuzzy sets are represented by type-2 membership values that are assigned a range of values rather than a crisp value. Based on the discussion, the type-2 fuzzy sets can be written mathematically as shown below.

$$\tilde{A} = \left\{ ((x,u), \mu_{\tilde{A}}(x,u) \mid \forall x \in X, \forall u \in J_x \subseteq [0,1] \right\} \qquad (5.1)$$

Where $0 \le \mu_{\tilde{A}}(x,u) \le 1$. This definition is particularly different from that of the type-1 fuzzy sets and can be derived as an extension of type-1 fuzzy sets shown in (5.2).

$$A = \left\{ x, \mu_{\tilde{A}}(x) \mid \forall x \in X \right\} \qquad (5.2)$$

Where $0 \le \mu_{\tilde{A}}(x) \le 1$. A type-2 fuzzy set has an additional dimension associated with its membership value $\mu_A(x)$. In simple words, for a type-1 fuzzy logic, when $x = x'$, the fuzzy membership function $\mu_A(x)$ has a single crisp value whereas for a type-2 fuzzy set, $\mu_A(x)$ would provide multiple values between the lower and upper bounds. A diagrammatic representation of type-2 fuzzy membership function is shown in Figure 5.2 and 5.3. The membership grade u is a crisp value when $J_{x'}$ is represented in a three dimensional space and $J_{x'}$ is the vertical slice of the membership function $\mu_{\tilde{A}}(x,u)$. The secondary membership function for the input $x = x'$ can be written as in (5.3).

$$\mu_{\tilde{A}}(x = x', u) \equiv \mu_{\tilde{A}}(x') = \int_{u \in J_x{}'} f_{x'}(u)/u \qquad (5.3)$$

Where $f_{x'}(u)$ is the amplitude of the secondary membership function and its value lies between [0,1] both included. This clearly indicates that there is a secondary value associated with $f_{x'}(u)$ for each primary grade.

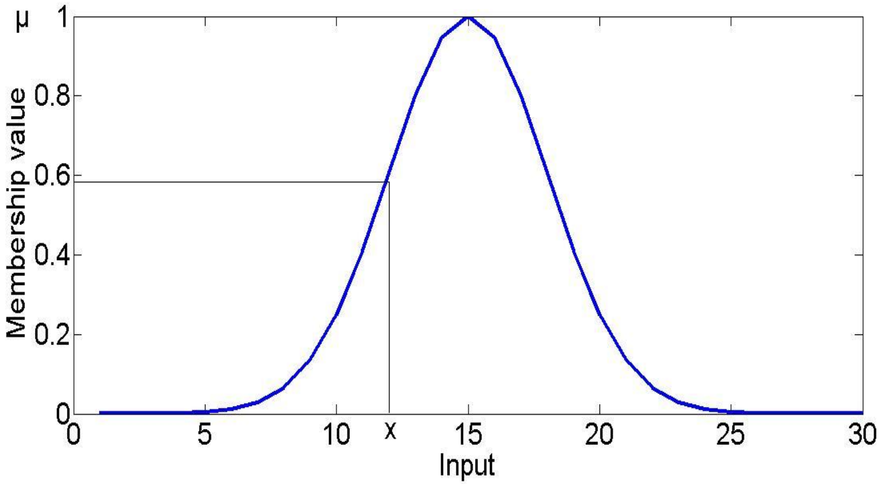


Figure 5.2. Type-1 Gaussian membership function

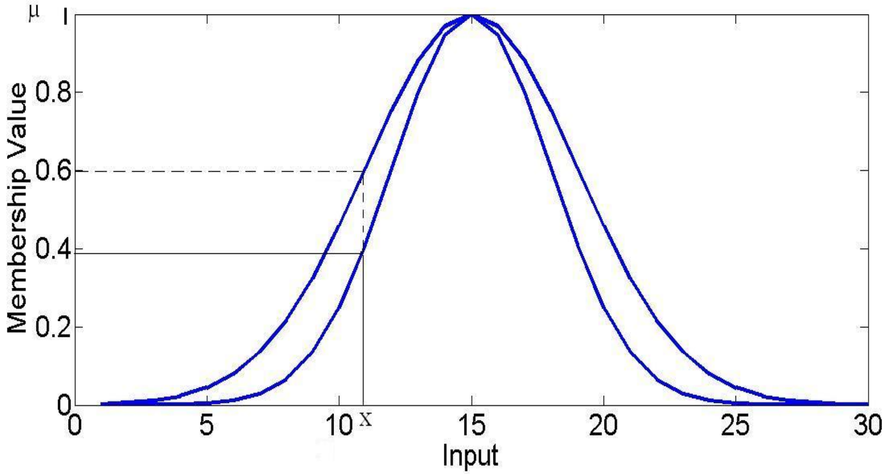


Figure 5.3. Type-2 fuzzy membership function with fixed mean and varying sigma values

The main problem with type-2 fuzzy logic in its generalized form is the amount of computation required to convert from type-2 fuzzy to type-1 fuzzy set before computing the crisp output value. The computational complexity is greatly reduced by using interval type-2 fuzzy sets[98], a special form of type-2 sets with the secondary membership function value set to unity. This assumption helps in the use of type-2 fuzzy sets for practical applications to a great extent. As all the input uncertainties are not known apriori, it is impossible to associate a proper secondary membership value to each input. Therefore the assumption of unity value for secondary membership grade while maintaining the property of type-2 fuzzy sets also reduces the computational requirements. Mathematically, interval type-2 fuzzy sets can be written as shown in (5.2) by modifying the general expression of type-2 fuzzy sets in (5.1).

$$\tilde{A} = \int_{x\in X} \int_{u\in J_x \subseteq [0,1]} 1/(x,u) = \int_{x\in X} \left[\int_{u\in J_x \subseteq [0,1]} 1/u \right] / x \qquad (5.4)$$

Where J_x is called the primary membership of *x*. The uncertainty about the fuzzy set is conveyed by the union all the primary memberships and is called as footprint of uncertainty [99].

The FOU is bounded by the upper and the lower membership function and can be represented as given in (5.5)

$$FOU(\tilde{A}) = \bigcup_{\forall x\in X} [\underline{\mu}_{\tilde{A}}(x), \overline{\mu}_{\tilde{A}}(x)] \qquad (5.5)$$

In the representation of the type-2 fuzzy sets, integration symbol is often used. This does not represent actual mathematical integration but represents continuous universe of discourse. For a discrete universe of discourse, summation symbol is used. Most of the real world applications, however, have a continuous universe of discourse, and it is a general practise to use the integration symbol.

The complexity as well as the uniqueness of type-2 fuzzy logic lies in the presence of FOU. The footprint of uncertainty is usually obtained by blurring the edges of the type-1 fuzzy membership function. It is possible to represent a type-2 fuzzy set using a large number of embedded type-1 fuzzy sets. The embedded fuzzy sets are formed such that at each input '*x*', only a single value of primary membership grade between the upper and lower range is used to construct the embedded set.

These embedded fuzzy sets increase computational requirements. Even if a computationally less expensive method like centroid or center of sets methods is used to obtain a crisp value, it incurs huge computational burden as it requires computing centroid values of large number of embedded type-1 fuzzy sets to obtain an average value. Reduction in the computational requirement has been achieved by using different methods like iterative procedure [100], uncertainty bounds [101] method or using geometric properties [102] of the membership function.

5.1.1. Union of fuzzy sets

Based on the set theoretic operators available, the union operator for type-2 fuzzy can be obtained according to the application requirement. T-conorm operator is usually used for fuzzy logic. The advantages of using T-conorm operator is given in [103] . Usually a maximum function is used for t-conorm operation to obtain the union of fuzzy sets. The union of two fuzzy sets is as shown in (5.6)

$$\mu_{\tilde{A}\cup\cdots\cup\tilde{z}}(x) = \int_{u\in J_x^1} \cdots\cdot \int_{w\in J_x^2} f_x(u) * \cdots\cdot * g_x(u) / v$$

$$\equiv \mu_{\tilde{A}}(x) \coprod \cdots\cdot \coprod \mu_{\tilde{z}}(x) \qquad x \in X \qquad (5.6)$$

Where $v \equiv u \vee ... \vee w$ and $\vee$ denotes the maximum t-conorm operator. The symbol * denotes minimum or product t-norm. The function of the union operator is to enumerate all the possible maximum values of the primary membership grades of the two sets and extract the minimum of the secondary membership grade.

5.1.2. Intersection of fuzzy sets

The set theoretic operator commonly used for performing the intersection operation is usually the product t-norm operator. The operation is performed as shown in (5.7).

$$\mu_{\tilde{A}\cap\cdots\cap\tilde{z}}(x) = \int_{u\in J_x^1} \cdots\cdots \int_{w\in J_x^2} f_x(u) * \cdots\cdots * g_x(u)/v$$

$$\equiv \mu_{\tilde{A}}(x)\prod\cdots\prod\mu_{\tilde{z}}(x) \qquad x\in X \qquad (5.7)$$

Where $v\equiv u\wedge...\wedge w$ and $\wedge$ denotes the product t-norm operator. The meet or intersection operator effectively enumerates the minimum value of the primary and the secondary membership grades of all the fuzzy sets.

5.1.3. Complement of fuzzy sets

The complement of fuzzy sets is got as the primary membership grade subtracted from unity. The representation of the complement operation is as shown in (5.8).

$$\mu_{\bar{\tilde{A}}}(x) = \int_{u\in J_x^u} f_x(u)/(1-u) \equiv \neg\mu_{\tilde{A}}(x) \qquad x\in X \qquad (5.8)$$

Where $\neg$ denotes negation operator.

5.1.4. Karnik Mendel [KM] algorithm for defuzzification

For performing defuzzification and to obtain a crisp value as output, different methods like centre of sets, centroid, centre-of-sums and height type-reduction can be used. Defuzzification in type-2 fuzzy sets requires computing the average value of multiple number of centroid values computed for all the probable embedded fuzzy sets that can be drawn. This is computationally intensive and prevents the use of type-2 fuzzy sets in real-world applications. To circumvent this limitation, Karnik and Mendel designed an iterative algorithm [100] that can obtain the crisp value in maximum of M iterations, where M is the maximum number of rules in the rule base.

This is achieved by computing the left and right end points. The end points refers to the point where the lower membership values shift to higher membership function values and vice versa.

The KM algorithm for computing y_r is :

1. Without any loss of generality, assume the pre-computed value of y_r^i are arranged in ascending order as shown $y_r^1 \leq y_r^2 \leq y_r^3 \leq \cdots \leq y_r^M$, where M is the number of rules
2. Compute the y_r as $y_r = \sum_{i=1}^{M} f_r^i y_r^i / \sum_{i=1}^{M} f_r^i$ by initially setting the right end point firing strength as $f_r^i = \left(\underline{f}^i + \overline{f^i}\right)/2$ for $i = 1 \ldots \ldots M$ and let $y_r' \equiv y_r$
3. Find R $(1 \leq R \leq M - 1)$ such that $y_r^R \leq y_r' \leq y_r^{R+1}$
4. Compute $y_r = \sum_{i=1}^{M} f_r^i y_r^i / \sum_{i=1}^{M} f_r^i$ with $f_r^i = \underline{f}^i$ for $i \leq R$ and $f_r^i = \overline{f^i}$ for $i > R$ and let $y_r'' \equiv y_r$
5. If $y_r'' \neq y_r'$ then go to Step 6 else stop and set $y_r'' \equiv y_r$
6. Set y_r^i equal to y_r'' and return to Step 3.

The same method is employed to compute the left-end point of the consequent. The defuzzified output of an IT2 fuzzy logic system can be calculated as the average of y_l and y_r as shown in (5.9)

$$y(X') = \big(y_l(X') + y_r(X')\big)/2 \qquad (5.9)$$

5.1.5. Geometric defuzzification

Geometric type-2 fuzzy set can be defined as a collection of polygons in three dimensional space with the edges forming the triangle [104]. In geometric defuzzification, the type-reducer and the defuzzifier are combined to form a single functional block, thereby reducing the computational requirements associated with the type-reducer, and to directly provide the crisp output from the type-2 fuzzy consequent sets. Centroid of the geometric type-2 fuzzy consequent can be calculated as the center of the geometric shape of the final consequent set obtained using the Bentley-Ottman plane sweep algorithm [105] after the calculation of firing levels of all the rules. The plane sweep can be performed by ordering the discretized values of the coordinate points in the consequent type-2 fuzzy set, and the corresponding centroid (5.10) can be calculated by finding the closed polygon formed by the edges of the ordered coordinates with the origin .

$$centroid = \frac{\sum_{i=0}^{n-1}\left(x_i + x_{i+1}\right)\left(x_i y_{i+1} - x_{i+1} y_i\right)}{3\sum_{i=0}^{n-1}\left(x_i y_{i+1} - x_{i+1} y_i\right)} \tag{5.10}$$

Figure 5.4 shows a consequent type-2 fuzzy set, where the lower and upper coordinates are arranged in proper ascending and descending order. The two different shaded regions in figure 5.4 indicate the triangle formed by using two adjacent points on consequent and the origin to construct the final closed polygon.

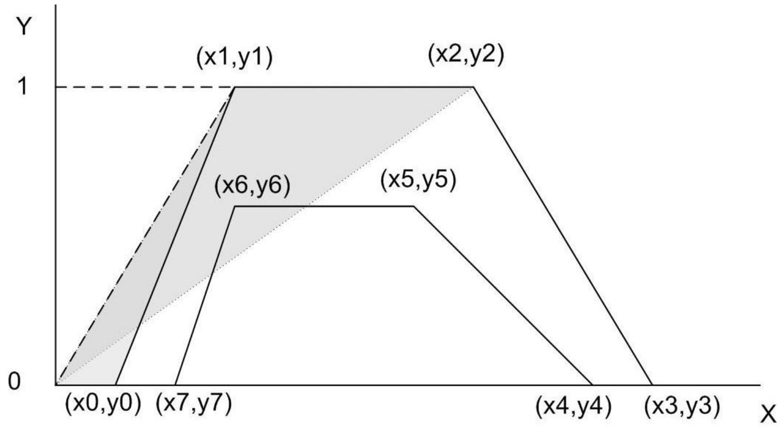


Figure 5.4. Ordered coordinates geometric consequent set showing two of the closed polygons

5.2. Appropriate situations for applying Type-2 FLS

The most appropriate situations under which Type-2 FLS performs the best are highlighted in [106]. They can be summarized as follows:

- Non-stationary noise associated with the sensor measurements that 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 fuzzy logic system is mined from a series of if-then questionnaires put forward to experts.

All these properties are existent in the urban vehicular traffic.

- The induction loop detectors used are easily affected by the prevalent external environmental conditions. This causes loss of data, improper detection of vehicles, improper classification of the length of the vehicles thereby leading to a calculation that predicts higher vehicle count than the actual value. This can be assumed as random noise added to the input data and varies according to the conditions of sensor and environment.
- The traffic release pattern is highly stochastic in nature. The exact number of vehicles that would enter a section of the network is not easily predictable as it depends to a great extent on the drivers' behaviour and their need for travel. Additionally, the presence of traffic signals produces a pseudo random effect.

This is mainly due to the formation of platoon of vehicles at the intersection because of the traffic signal at intersections.

- Traffic at an intersection can be easily defined using linguistic rules. It is much easier to define the control sequence as a set of rules. The rules are usually constructed using expert opinion. These rules are prone to error with increase in the number of links, lanes and phases at an intersection.

All of the above mentioned properties of the traffic system makes it an ideal candidate for type-2 fuzzy system based control.

5.3. CLASSIFICATION OF THE PROPOSED DECISION SYSTEMS

Four different types of decision systems are proposed in this work. The first two decision systems (T2DR and GFMAS) are designed based on heuristics and deductive reasoning. Third decision system (SET2) design is based on a stochastic symbiotic evolutionary learning approach to adapt the parameters of the type-2 fuzzy system. This decision system uses an online batch learning approach. The fourth decision system designed is a Q-learning based neuro-type2 fuzzy system (QLT2). It is an online adaptive system where the parameters of the neuro-fuzzy system are adapted using back propagation technique using error value computed with reinforcement Q-learning technique.

5.4. TYPE-2 FUZZY DEDUCTIVE REASONING DECISION SYSTEM

The first heuristics based type-2 fuzzy decision system designed is T2DR (Type-2 fuzzy Deductive reasoning) system. The block diagram of the decision system is shown in Figure 5.5 and details of the designed decision system are described in the following sections.

5.4.1. Traffic data inputs and fuzzy rule base

T2DR decision system uses the *flow, queue* and communicated *neighbour status data* to compute the green time. The data collection module provides the necessary data to the decision system. First step performed inside the decision module is the fuzzification of crisp inputs received from the data collection module. The *queue* and *flow* data are clustered into three regions and the universe of discourse of the inputs are decided using the *hotspot* and saturation *flow* rate parameters. *Hotspot* refers to the condition of the link that experiences a queue length greater than threshold number of vehicles beyond which link can be classified as in high congestion state. A vehicle count of thirteen was used as a threshold value to classify the link as hotspot [62]. *Saturation flow* rate refers to the equivalent hourly rate at which vehicles can traverse an intersection under prevailing conditions, assuming a constant green indication at all time and no loss time, expressed in vehicles per hour or vehicles per hour per lane. A *saturation flow* rate of 2400 vehicle per hour was used to normalize the flow data.

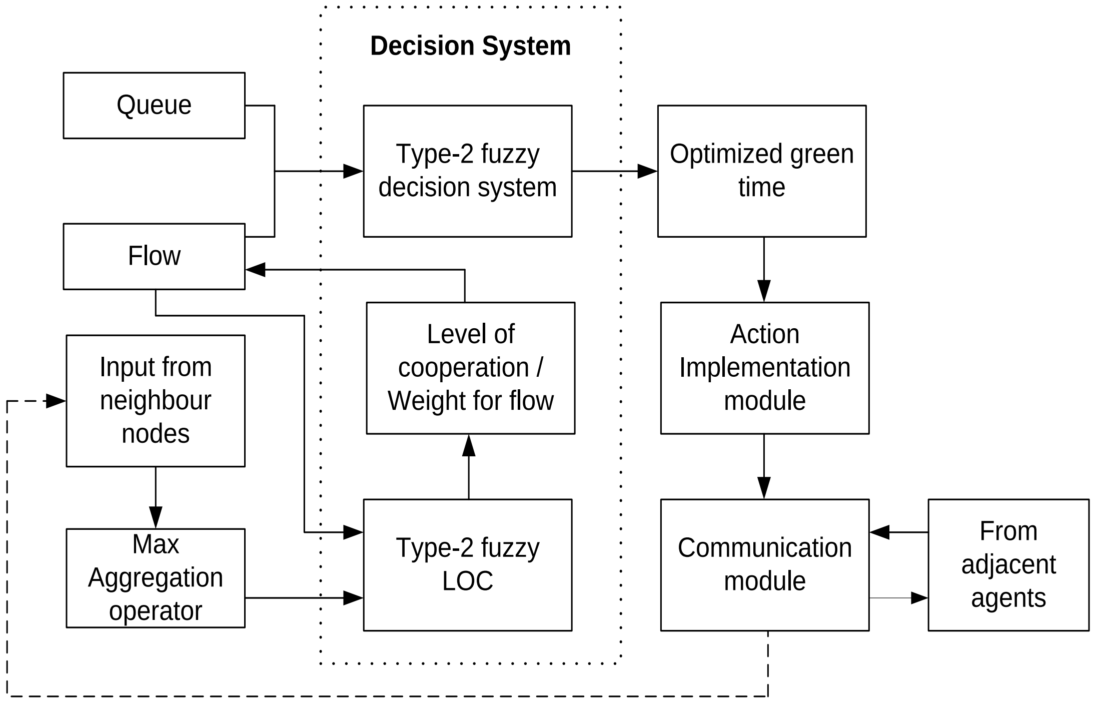


Figure 5.5. Block diagram of T2DR multi-agent weighted input decision system

Membership grade for each of the clustered input was assigned using a Gaussian function with overlapping regions. The mean values of the Gaussian functions represent the centre of the cluster formed and the values were chosen to be closer to the values used in [62]. The upper and lower boundary standard deviation values were calculated based on the *flow* rates and *queues* experienced at different time periods and same period of the day in a week. The fuzzified antecedents and consequents of the fuzzy rules are as shown in Table 5.1 and 5.2.

The consequents were designed as type-2 fuzzy sets to account for the uncertainties associated with scaling factor and the signal timings. In this work, type-2 membership functions were developed based on heuristics and type-1 fuzzy membership functions used previously in [45]. The type-1 fuzzy membership function in [44] was used as the base and blurred to create lower and upper membership grade. The details of antecedent and consequent membership functions used are shown in Figure 5.6. The maximum green time allocated for a phase is fixed at 60 seconds even though the recommended value is 40 seconds to provide more degree of freedom for the signal control and a minimum green time of 10 seconds to avoid short phase, switching loss and increased travel time delay.

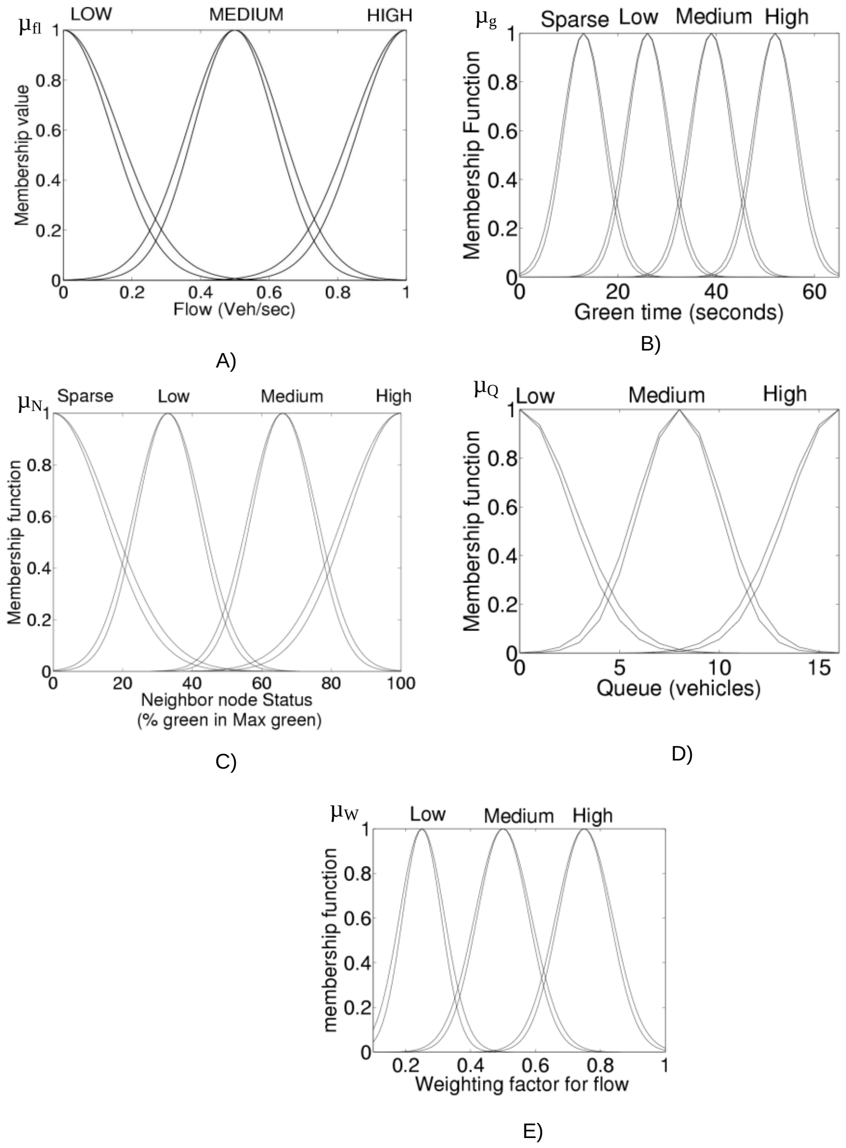


Figure 5.6. Antecedent and consequent fuzzy membership functions

Selection of appropriate rules and mapping of the antecedents to consequents is an important objective. In general practice, the rules are created by trial and error based on expert knowledge. Type-2 fuzzy system produces better results than type-1 fuzzy system under these conditions [106]. The structure of the developed rule base is as shown

- ✓ Level of cooperation decision fuzzy system - **If {*flow is high*} and {*Neighbor status is low*} then {*Level of cooperation is low*}**
- ✓ Green time decision fuzzy system - **If {*Queue is low*} and {*flow is high*} then {*traffic is low*}**

Table 5.1. Mapping of flow and Neighbour state inputs to consequent weighting factor output

		flow		
		low	Medium	High
Neighbour state	sparse	low	low	low
	low	low	low	low
	Medium	low	medium	medium
	High	medium	high	high

Table 5.2. Mapping of flow and queue input to consequent green time output

		Flow		
		low	Medium	High
Queue	low	sparse	low	low
	Medium	high	medium	medium
	High	high	high	high

Table 5.1 and 5.2 show the rule base for both the belief or level of cooperation and the fuzzy decision module. The initial rule base was designed based on the understanding of [107] in creating the rule base from data. The rule base was fine-tuned and validated using a small section of the network with six intersections. The initial origin-destination matrix or the traffic demand was modified from the original matrix created from actual data to avoid saturation of the network as suggested in [108]. The rules were initially modelled for fuzzy decision module without any communication between the agents. The validation of the rule base was performed by comparing the total mean delay experienced by a vehicle inside a network to HMS traffic signal controller. A detailed explanation of Hierarchical Multi-agent System (HMS) was provided in chapter 3 . The rule base with the smallest travel time delay in comparison to HMS was used in this work. The same procedure was repeated for the agent system with communication capabilities to develop the belief model which decides the level of cooperation. The initial rule base was developed based on intuition and basic understanding of the relationship between traffic *flow* and *queue*.

5.4.2. Inference Engine

Inference is the key component of a fuzzy system. The antecedents membership functions are used to calculate the firing level for each rule in the rule base and is applied to the consequent fuzzy sets. The outputs from the belief and fuzzy decision system for the l-th fuzzy rule can be written as in (5.11) and (5.13).

$$\mu_{B1^l}(y) = \mu_{w^l}(y) \sqcap \left\{ \begin{array}{l} \left[\sqcup_{x1 \in fl} \left\{ \mu_{fl^l}(x1) \right\} \sqcap \mu_{F_1^l}(y) \right] \\ \sqcap \left[\sqcup_{x2 \in N} \left\{ \mu_{N^l}(x2) \right\} \sqcap \mu_{F_2^l}(y) \right] \end{array} \right\} \forall\, y \in w \qquad (5.11)$$

$$\mu_{wfl}(x) = w * e[-0.5 * ((x_{mean} - x)/\sigma)^2] \qquad \forall\, x \in flow \qquad (5.12)$$

The weight parameter for the *flow* input was calculated based on the instantaneous *flow* value and the communicated *neighbour status* data. Max function was used to aggregate the communicated *neighbour status data* to enable coordination with respect to most congested neighbour. The computed weight parameter calculated is inversely proportional to the level of cooperation among the agents. A large weight value causes the *flow* input to the decision making fuzzy system of the inference engine to have a higher membership value as shown in (5.12). This makes the output of the fuzzy system to be influenced by the queue input and less dependent on neighbouring agents.

The *flow* input was used for deciding the level of cooperation, as it is an averaged value giving the overall traffic situation of the neighbours and calculated from upstream, downstream and queue spill back inputs. The instantaneous value of *queue* was determined using ‘max’ function as they indicate current traffic condition at an intersection and ensure the green time calculated is with respect to maximum congested lane.

$$\mu_{B2^l}(y) = \mu_{g^l}(y) \sqcap \left\{ \begin{array}{c} \left[\sqcup_{x1\in q}\left\{\mu_{q^l}(x1)\right\} \sqcap \mu_{F_1^l}(y)\right] \\ \sqcap \left[\sqcup_{x2\in fl}\left\{\mu_{wfl^l}(x2)\right\} \sqcap \mu_{F_2^l}(y)\right] \end{array} \right\} \ \forall\, y \ \in g \qquad (5.13)$$

Where $\mu_{B1^l}(y)$, $\mu_{B2^l}(y)$ are the outputs of the fuzzy systems, $\mu_{F_1^l}(y)$, $\mu_{F_2^l}(y)$ are the type-2 membership functions for the antecedent inputs for the l-th rule.

$\mu_{fl^l}(x1)$, $\mu_{q^l}(x1)$, $\mu_{wfl^l}(x2)$, $\mu_{N^l}(x2)$ are the type-2 fuzzy membership functions of the inputs. Since the interval type-2 fuzzy system is used for determining the optimized green time, the firing level obtained will also be expressed as an interval set:

$$F^l(x) = \left[\underline{f}^l(x), \overline{f}^l(x)\right] \equiv \left[\underline{f}^l, \overline{f}^l\right] \qquad (5.14)$$

The lower and the upper bounds of the level of cooperation deciding type-2 fuzzy system of the inference engine can be written as in (5.15) and (5.16) respectively.

$$\underline{f}^l(x) = \underbrace{sup}_{x} \int_{x1\in N}^{\infty} \int_{x2\in fl}^{\infty} \left[\underline{\mu}_N(x1) * \ \underline{\mu}_{F_1^l}(x1)\right] * \left[\underline{\mu}_{fl}(x2) * \ \underline{\mu}_{F_2^l}(x2)\right] / x \qquad (5.15)$$

$$\overline{f}^l(x) = \underbrace{sup}_{x} \int_{x1\in N}^{\infty} \int_{x2\in fl}^{\infty} \left[\overline{\mu}_N(x1) * \ \overline{\mu}_{F_1^l}(x1)\right] * \left[\overline{\mu}_{fl}(x2) * \ \overline{\mu}_{F_2^l}(x2)\right] / x \qquad (5.16)$$

In a similar fashion, the lower and upper bounds of the decision making fuzzy system can be written as in (5.17) and (5.18):

$$\underline{f}^l(x) = \underbrace{sup}_{x} \int_{x1\in q}^{\infty} \int_{x2\in wfl}^{\infty} \left[\underline{\mu}_q(x1) * \ \underline{\mu}_{F_1^l}(x1)\right] * \left[\underline{\mu}_{wfl}(x2) * \ \underline{\mu}_{F_2^l}(x2)\right] / x \qquad (5.17)$$

$$\overline{f}^{l}(x) = \underbrace{sup}_{x} \int_{x1 \in q}^{\infty} \int_{x2 \in wfl}^{\infty} \left[\overline{\mu}_{q}(x1) * \overline{\mu}_{wF_1^l}(x1) \right] * \left[\overline{\mu}_{wfl}(x2) * \overline{\mu}_{F_2^l}(x2) \right] / x \qquad (5.18)$$

Output from the fuzzy system for all the rules can be aggregated together as shown in (5.19) and (5.20)

$$\mu_{B1}(y) = \coprod_{l=1}^{m} \mu_{B1^l}(y) \qquad (5.19)$$

$$\mu_{B1}(y) = \coprod_{l=1}^{m} \mu_{B1^l}(y) \qquad (5.20)$$

Where ‘m’ refers to the number of rules in the decision making type-2 fuzzy system. Karnik-Mendel algorithm was used to convert the type-2 fuzzy outputs into type-1 fuzzy output. The crisp output is calculated as average value of the right and left end points calculated using the equations given in (5.21) and (5.22) respectively.

$$y_r = {\sum_{i=1}^{M} \{ f_r^i y_r^i \}} \Big/ {\sum_{i=1}^{M} f_r^i} \qquad (5.21)$$

$$y_l = {\sum_{i=1}^{M} \{ f_l^i y_l^i \}} \Big/ {\sum_{i=1}^{M} f_l^i} \qquad (5.22)$$

$$y = \frac{y_r + y_l}{2} \qquad (5.23)$$

The main objective of the firing level is to minimize the lower bound and maximize the upper bound. As the number of rules were restricted to a fixed value, the number of iterations required to compute the left and right end points is bound within the fixed value.

The final green value output selected is applied to the traffic signal through action implementation module. The action implementation module has an additional task of checking for the cycle time and adjusting the values proportionally for each phase, as maximum cycle length allowed is restricted to 120 seconds.

5.5. GEOMETRIC FUZZY MULTI-AGENT SYSTEM

The computational complexity associated with the T2DR (Type-2 fuzzy deductive reasoning) decision system and also the difficulty in computing the associated weight for the input necessitates the development of lesser complex system with internal belief model and trapezoidal membership function for ease of implementing a lesser complex defuzzification process developed based on the geometric properties. It was also found that not all the inputs and output need to be of type-2 fuzzy set. To satisfy these requirements a geometric type-2 fuzzy decision process was developed and explained in detail in the following sections.

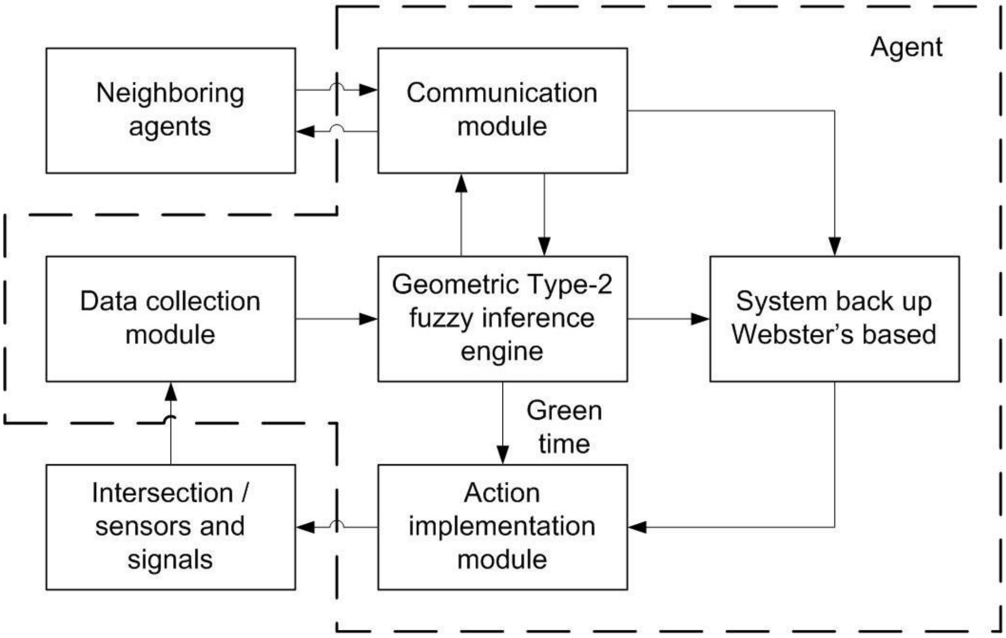

Agent
Neighboring agents
Communication module
Data collection module
Geometric Type-2 fuzzy inference engine
System back up Webster's based
Green time
Intersection / sensors and signals
Action implementation module

Figure 5.7. GFMAS agent architecture

5.5.1. Input Fuzzifier

The averaged values of *flow* and *queue* data collected from the road network and communicated data is passed on to the geometric type-2 inference system for calculating the green time required for the phase during the next cycle period. Geometric inference system has the same functional blocks as those of a type-2 fuzzy system, except the type-reducer and defuzzifier are merged into a single block called geometric defuzzifier as shown in Figure 5.8. The initial step in the inference system is the fuzzification of the inputs, in which a measure of possibility is assigned to each input. Since the fuzzy set used was an interval type-2 fuzzy set, the inputs were assigned upper and lower membership grades or measure of possibility based on their membership functions, and the secondary membership grade associated with each primary membership grade assigned a value of unity. The union of all the membership grades between the upper and lower bounds would give the foot print of uncertainty of the input in the classified region. The membership functions were designed in trapezoidal shape and divided into three regions.

The lower and upper bounds in each region were decided based on the maximum and minimum flow rate experienced at an intersection during the specified time period of weekdays thereby, each input is associated with a range of membership grade rather than a crisp point, thus retaining the fuzziness. In a similar manner, the queue count was designed as Type-2 fuzzy set, and each input was assigned a boundary membership grade. The final input used was the data received from the communication module which is a Type-1 fuzzy set. The fuzzified inputs are shown in Figure 5.9.

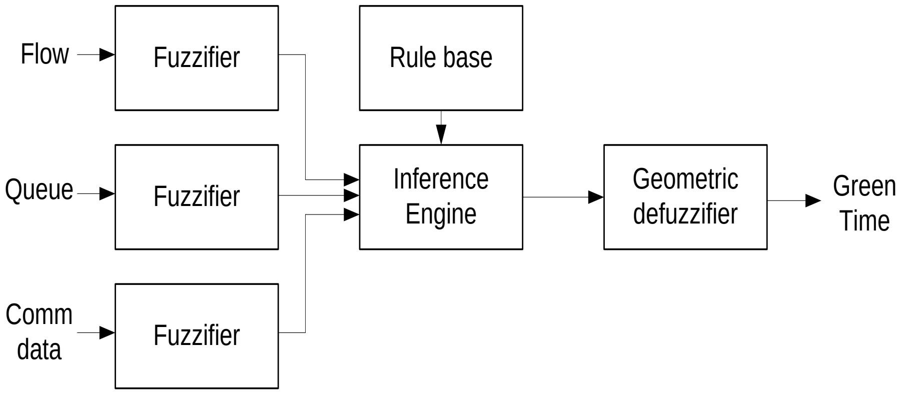


Figure 5.8. Block diagram of geometric type-2 fuzzy system

The communication module calculates the percentage of green time allocated to a phase by the signal with respect to the maximum permissible green time. Green time can serve as a direct indication of the traffic state at an intersection if the value is calculated dynamically based on the traffic data and the traffic flowing in the outgoing links. However, the relationship becomes highly non-linear due to the influence of traffic in the incoming links and platoon formation. The level of uncertainty associated with green timing in a phase can be easily handled by using type-1 fuzzy sets itself as the only source of uncertainty is the communication noise.

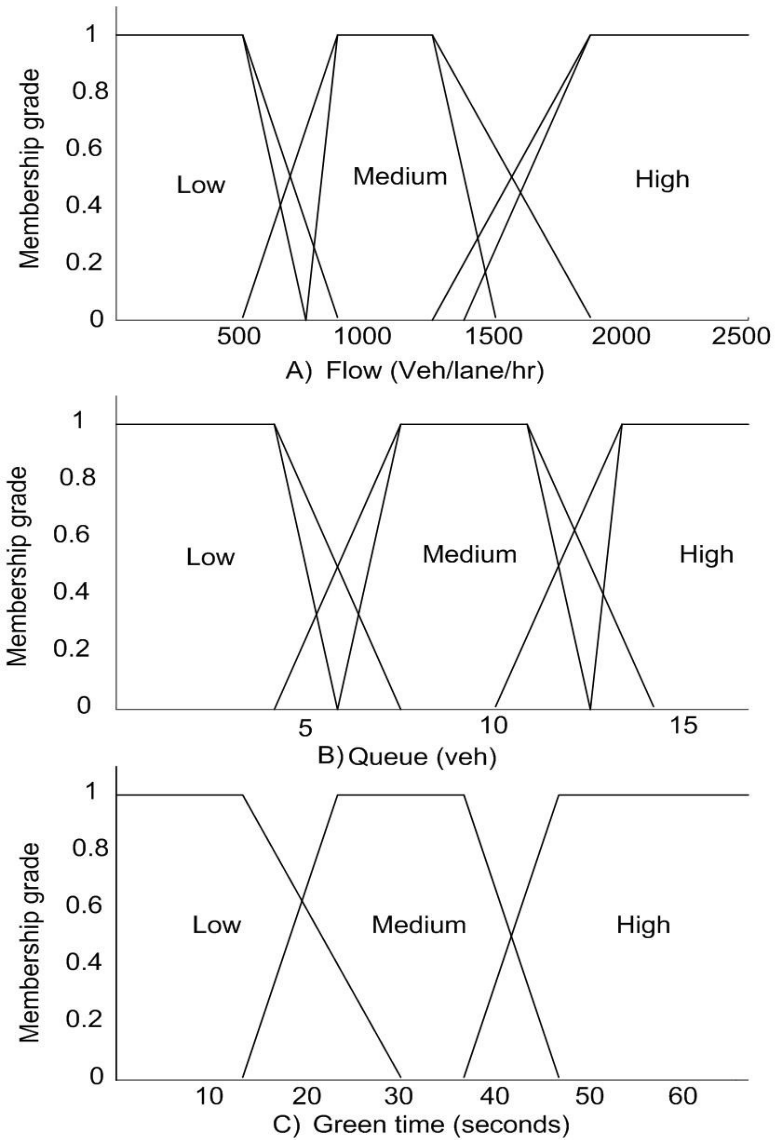


Figure 5.9. Fuzzified antecedents and consequents in a GFMAS

The green timing calculated solely based on the local traffic data would cause increased congestion levels in the outgoing links. This can cause already congested neighbouring intersections to experience even higher inflow of vehicles, and extension of queue in the link beyond the incoming intersection, resulting in deadlocks. Therefore coordination based on communicated congestion data becomes essential during the periods of heavy and medium traffic flow. Reduced value of green time is allocated to a specific phase if the intersection connected to the outgoing link is already congested. This prevents queue spill back and deadlock formation. This coordination is achieved based on the rule base information and highly congested neighbour communicated data.

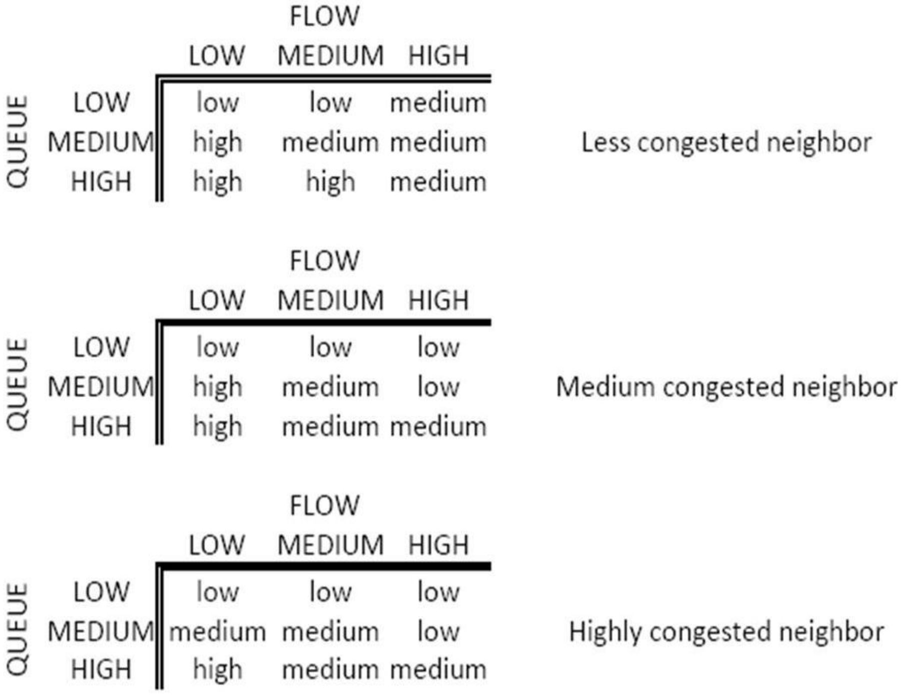
Since the communicated congestion data is similar to the consequent green time except for their representation as percentage, both have a similar classification. The membership functions are designed by dividing into three equal regions with the overlap calculated based on Webster's equation. As the coordination is with respect to the maximally congested neighbour, maximum value of the communicated congestion data from the neighbouring intersections is used as input. The communication module in short performs the functions of data reception, transmission of congestion status to neighbours and data mining operation.

5.5.2. Inference engine

Inference engine is the core of a multi-agent system. The inputs and outputs to the type-2 fuzzy inference engine are shown in Figure 5.9. The inference engine calculates the lower and upper threshold of firing levels for each rule in the rule base. A total of twenty seven rules were created based on the three inputs as shown in Figure 5.10. When the communicated congestion data indicates low traffic congestion in the neighbouring intersections, greedy signal policy was used to calculate the green time with minimal cooperation. For adjacent intersections with medium and high traffic congestion levels, the signal policy was suitably modified to produce smaller duration of green time than with greedy signal policy. This ensures smaller inflow of vehicles into the already congested intersection, enabling the intersection to clear its traffic faster.

The structure of the rule base is as shown below.

"If Flow input is in Low region and Queue input is in High region and Communicated data is in Low region then Green time is in High region"



	FLOW		
QUEUE	LOW	MEDIUM	HIGH
LOW	low	low	medium
MEDIUM	high	medium	medium
HIGH	high	high	medium

Less congested neighbor

	FLOW		
QUEUE	LOW	MEDIUM	HIGH
LOW	low	low	low
MEDIUM	high	medium	low
HIGH	high	medium	medium

Medium congested neighbor

	FLOW		
QUEUE	LOW	MEDIUM	HIGH
LOW	low	low	low
MEDIUM	medium	medium	low
HIGH	high	medium	medium

Highly congested neighbor

Figure 5.10. Rule base for the GFMAS signal control

The 'AND' operation is performed by using the t-norm (minimum function). Since the geometric inference system used was a Type-2 fuzzy system, the lower and upper firing threshold values for the consequent set for each rule were calculated using (5.24) and (5.25) respectively

$$\underline{f}^{l}(x) = \underbrace{sup}_{x} \int_{x1\in q}^{\infty}\int_{x2\in fl}^{\infty}\int_{x3\in cd}^{\infty} \left[\underline{\mu}_{q}(x1) * \underline{\mu}_{F_1^l}(x1)\right] * \left[\underline{\mu}_{fl}(x2) * \underline{\mu}_{F_2^l}(x2)\right] * \left[\underline{\mu}_{cd}(x3) * \underline{\mu}_{F_3^l}(x3)\right]/x \qquad (5.24)$$

$$\overline{f}^{l}(x) = \underbrace{sup}_{x} \int_{x1\in q}^{\infty}\int_{x2\in fl}^{\infty}\int_{x3\in cd}^{\infty} \left[\overline{\mu}_{q}(x1) * \overline{\mu}_{F_1^l}(x1)\right] * \left[\overline{\mu}_{fl}(x2) * \overline{\mu}_{F_2^l}(x2)\right] * \left[\overline{\mu}_{cd}(x3) * \overline{\mu}_{F_3^l}(x3)\right]/x \qquad (5.25)$$

Where $\underline{\mu}_{F_1^l}(x1)$, $\underline{\mu}_{F_2^l}(x2)$ and $\underline{\mu}_{F_3^l}(x3)$ are the lower membership function, $\overline{\mu}_{F_1^l}(x1), \overline{\mu}_{F_2^l}(x2), \overline{\mu}_{F_3^l}(x3)$ are the upper membership functions of antecedents inputs for the l-th rule.

$\left[\left(\underline{\mu}_{q}(x1), \underline{\mu}_{fl}(x2), \underline{\mu}_{cd}(x3)\right), \left(\overline{\mu}_{q}(x1), \overline{\mu}_{fl}(x2), \overline{\mu}_{cd}(x3)\right)\right]$ are the lower and upper membership functions of the inputs queue, flow and communicated congestion data respectively. The supremum value is attained when the terms inside the bracket attain the least upper bound value. The firing levels of each rule is used to clip the consequents membership function and derive the output type-2 fuzzy set.

Geometric defuzzification is used to calculate the crisp value of the green time from the derived output type-2 fuzzy set. Consequents geometrical properties are used to derive calculate the centroid of the shape. This requires ordering of the discretized lower and upper membership functions in ascending and descending order respectively. The coordinates are obtained using the plane sweeping algorithm that gives the non-overlapping edges of the trapezoid and the point of intersection of different regions of green time or the consequent fuzzy set. Figure 5.11 shows the process of arranging the coordinate points and also removing overlap regions.

Once sorted, triangles can be constructed by using two adjacent points on the consequent set and the origin. The resultant average value of the centroid of all the triangles constructed on the consequent set meets the geometric centre of gravity of the output type-2 fuzzy set which is the closed polygon formed by connecting all edges to origin.

For better accuracy of the calculated value of centroid, it is essential to discretize the consequent set into a large number of points to increase the number of overlapping polygons used to average the centroid value. The trade-off would be the computational cost associated with the process. The signal control decisions are based on traffic in the incoming and outgoing links, but not on the traffic in lanes without right of way. Since the green time for each phase is calculated online at the end of each phase, it is difficult to consider the competing phase timings.

However, the maximum green time limitation imposed on each phase minimizes the possibility of indiscriminate increase in green time for a phase. The signal control also begins to allocate more time to a phase in a recursive manner once a queue build up is detected. It can be said that the traffic at lanes without right of way is taken care of with a delay of one cycle period.

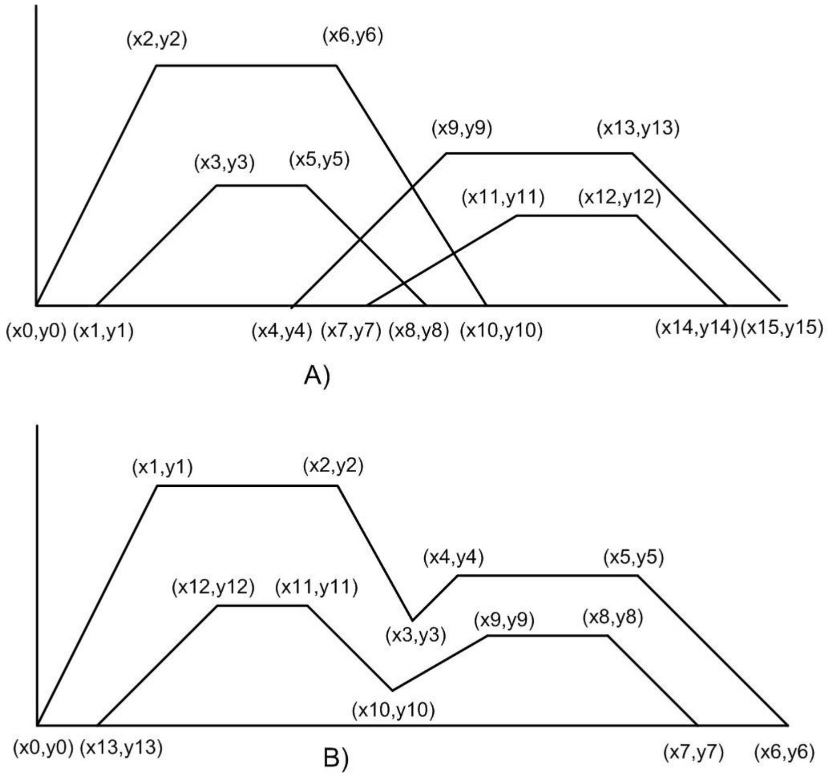


Figure 5.11. Geometric defuzzification process based on Bentley-Ottmann plane sweeping algorithm

5.6. SYMBIOTIC EVOLUTIONARY TYPE-2 FUZZY DECISION SYSTEM

SET2 stands for symbiotic evolutionary type-2 fuzzy decision system. In the previous sections, the rule base and the parameters for the creation of the input fuzzifier of the type-2 fuzzy system in both T2DR and GFMAS were designed based on the heuristics and the historical traffic data that were obtained when using a fixed-time signal control. The traffic data of flow and queue therefore does not truly represent the dynamics of the environment and does not account for the effect of the variation in signal timing in other connected intersections. To reduce the errors, the decision system utilized the communicated information of ratio of green time allocated by each phase in the adjacent intersection with respect to the maximum permissible value. However, this creates over dependency of the decision system on communicated information. Any absence of the data due to communication failure would affect the decisions taken using the constructed rule base in which communicated data is an integral part. This would trigger the use of backup fixed time plan until the communication is fixed and adversely affecting the performance of the entire system. In order to avoid this, it is essential to use the communicated data to build the model of the system than can effectively take care of the variations in the decisions of adjacent agents.

In order to address the issues mentioned above, SET2 decision system uses only the flow and queue input collected from the road network. Apart from these two inputs, rate of change in flow between two consecutive evaluation period is also used. This input is made possible by storing the values of average flow experienced during each phase in the previous cycle in the knowledge base and data repository module. This input gives an indication of the increase or decrease in the value of flow during the evaluation period. The universe of discourse for this particular input is normalized between [-1,1] using the saturation flow rate. A value of -1 would indicate a large decrease in the value of flow and literally would mean no vehicle crossed the intersection during the evaluation period and a value of +1 would indicate a large increase in the vehicle count from the previous evaluation period. The rate of change in flow input is therefore clustered into three clusters namely {*-ve, normal, +ve*}. The rate of change in flow is determined from the flow input received from the loop detectors placed at the intersection. Therefore, the rate of change in flow also has all the uncertainties that were associated with the flow input and is represented using type-2 fuzzy sets.

In the previous approaches, the parameters of the fuzzy sets and the rule base were generated based on the heuristic data. In order to make the decision system much more responsive to the dynamics of the traffic environment, all the parameters in SET2 are evolved using online simulation of the network. A batch mode online learning is employed, where a full set of training data created dynamically with online simulation is used for the training. Stochastic search based method like traditional GA can be used for evolving the parameters. However, traditional GA methods suffer from lack of diversity and requires the use of extremely high value of mutation rate to maintain the diversity but ends up losing on the exploitation front [109].

Use of a single chromosome to represent the entire solution presents a difficulty in evolving all the parameters and solution space search becomes too large to handle. Therefore a cooperative co-evolutionary approach is employed to generate the full solution. In the following section, an introduction to the basics of symbiotic evolutionary approach used to generate the decision system is presented.

5.6.1. Symbiotic evolution

Symbiotic evolution can be defined as a type of co-evolution process where different individuals explicitly cooperate with each other and rely on the presence of the other individual for their existence[110]. The symbiotic evolution is distinctly different from the co-evolutionary GA's. Most of the co-evolutionary process is developed based on the immune system or fitness sharing[111-113]. The immune algorithm is a global search algorithm that is based on the characteristics of the biological immune system. It is an adaptive, distributed, and parallel intelligent system that has the capability to control a complex system. The immune system is aimed at protecting living bodies from the invasion of various foreign substances called antigens, and eliminating them. When an antigen is introduced in a biological species, a specific antibody that is capable of detecting the antigen and eliminating it needs to be produced. This is achieved by forming a group of antibodies that detects maximum number of antigens introduced. Therefore, the antibodies needs to compete with each other to provide the desired result. The principles of immune system can be easily adopted to genetic algorithm with slight modification to the parent chromosome. In the modified GA, the chromosome represents partial solution instead of the complete solution . A combination of large number of these partial solutions would then produce the final antibody that maximizes the objective function. Here each partial solution compete with each other instead of cooperating with each other. Figure 5.12 Shows the solution generated by combining a large number of partial solutions. The partial solutions are referred to as "*Specializations*"

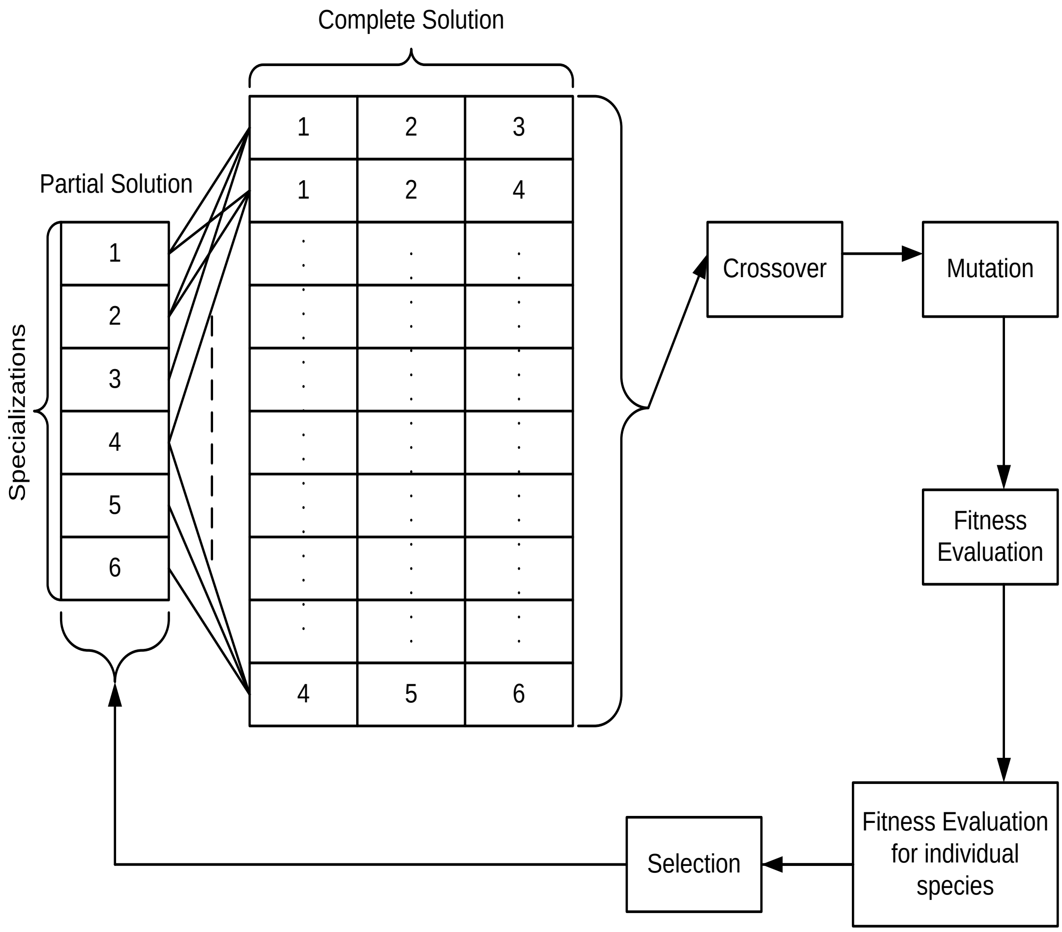


Figure 5.12. Block diagram of symbiotic evolution complete solution obtained by combining partial solutions

This is particularly not useful in large number of practical applications. For example, evolving an entire rule base of the fuzzy system can be cumbersome. When individual rules are created as species and combined together to form a full solution, the species should not be competing with each other but must cooperate with each other to obtain the optimal final solution.

When a single set of partial solution or specialization is used to create the final solution, the diversity can be affected to a great extent and large number of specializations would get repeated. To avoid this, an approach based on creating a large number of islands of specialization was introduced in [114]. Each island maintains a certain type of specialization that are unique to it and final solution is obtained by combining specializations gathered from each of the island. Figure 5.13 Shows the block diagram of one such islanded evolutionary approach. In [115-117], the fuzzy rule base was evolved using the islanded approach. Here each of the specialization represents a single rule - the parameters of each of the inputs are encoded to form the partial solution. The number of rules needs to be pre-fixed and each rule would represent a cluster in the solution space. It is possible that the solution evolved does not cover the entire solution space if the training data is not continuous and do not cover the entire state space. Therefore a modified version of the islanded symbiotic evolution is proposed in this work.

5.6.2. Proposed symbiotic evolutionary GA decision system

In the proposed symbiotic evolutionary genetic algorithm, instead of creating each rule along with its input parameters as a specialization, two islands were created. The first island represents the membership function values such as mean, lower and upper sigma values for the Gaussian membership function, and the second island represents the rule. A combination of the membership function and the corresponding rule would create the final solution. The solution set obtained can then be evaluated as done in traditional GA. Figure 5.14 Shows the diagrammatic representation of creating full solution using the proposed symbiotic evolutionary approach.

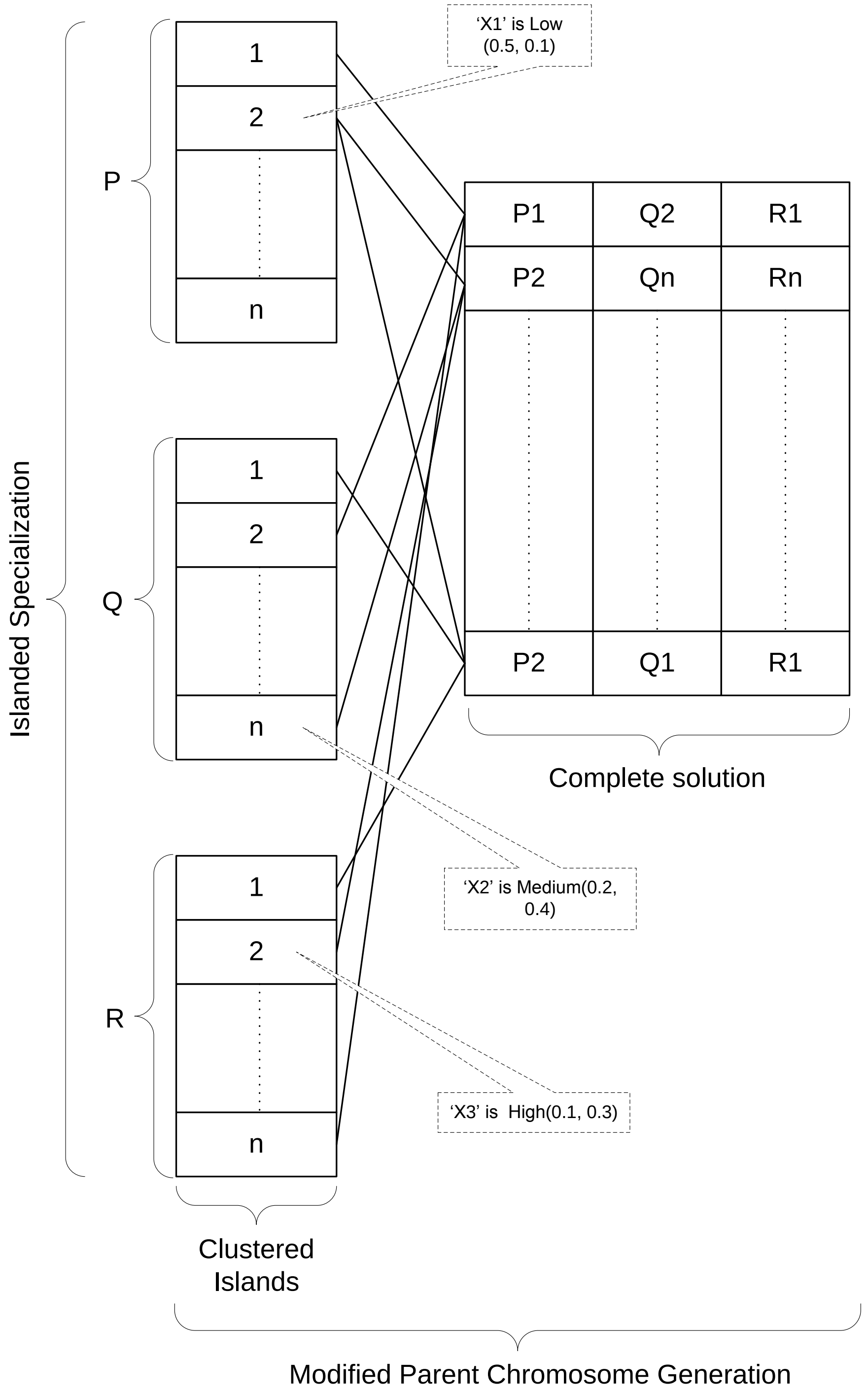


Figure 5.13. A representation of the islanded symbiotic evolutionary algorithm population

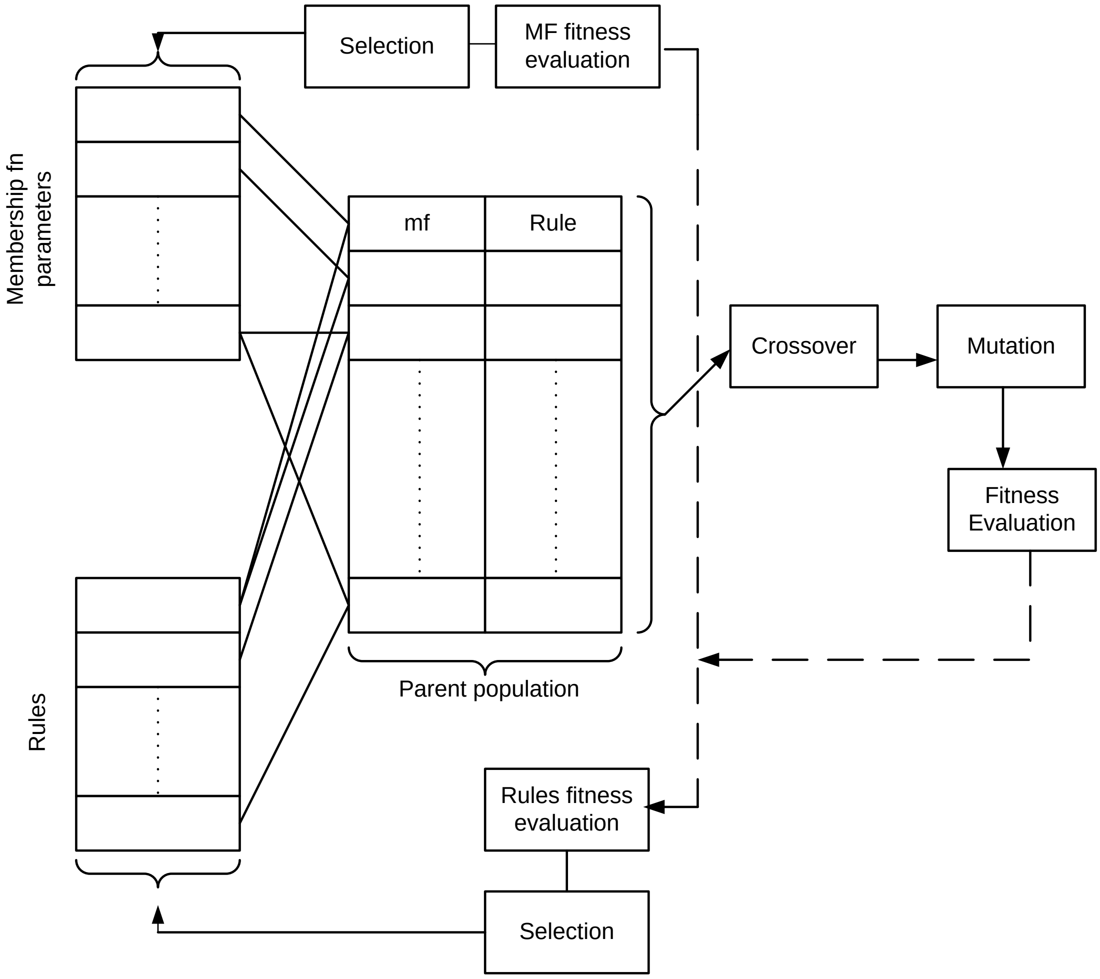


Figure 5.14. A block diagram representation of the symbiotic evolution in the proposed symbiotic evolutionary genetic algorithm

The values of membership function have continuous state space and hence real coded GA was used. However, binary coded GA was used for evolving the rule base as its chromosome alleles takes only two values (either active or inactive). The structure of the chromosome used in membership function island and the rule island specialization is shown in Figure 5.15 and Figure 5.16.

The parameters evolved is bounded within [0,1]. Real-coded GA ensures a continuous state space representation and avoids errors arising from discretization of state space.

The islanded rule specialization are binary coded and a value of one represents that the particular input is active and will be used in the computation of output. The cluster in the consequent that needs to be used in a particular rule is also evolved. The consequent part is represented using two bits. A decimal equivalent zero value indicates that the rules is invalid and would not be used. A value of one indicates first cluster, two represents second cluster and three represents the third cluster from the consequent to be used for the specific rule. Therefore the evolution of the rule base island would adapt the number of rules used in a particular evaluation.

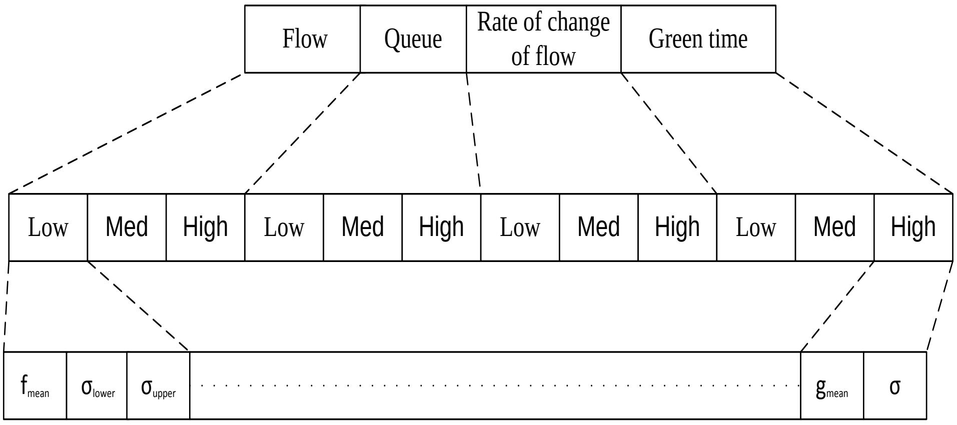


Figure 5.15. Structure of the chromosome for membership function cluster island

Obtaining the fitness value for the individual specialization is an extremely difficult task. The two sets of islanded clusters are dependent on each other and the fitness value is shared among each other. The fitness of each individual specialization is obtained as the average value of the final solution in which the particular specialization is used. Apart from this shared value of the fitness, it is also possible to use other parameters apart from the shared fitness to evaluate the partial solution.

The fitness function for the full population is computed in terms of the time delay value using the Webster’s formula introduced in Chapter 3. Apart from the delay value computed for the local intersection, the weighted communicated delay information from the neighbouring agents is also used for computing the fitness value. Equation (5.26) shows the final fitness function used.

$$Fitness(p) = \frac{1}{a}\sum_{k=1}^{a}\left(D_k + \sum_{i=1}^{in}\gamma_i * d_{ik} + \sum_{j=1}^{out}\beta_j * d_{jk}\right) \tag{5.26}$$

$$\gamma_i = No\ of\ phases\ vehicle\ released\ towards\ agent/No\ of\ phases$$

$$\beta_i = No\ of\ phases\ vehicle\ is\ received\ from\ the\ agent/No\ of\ phases$$

Where $Fitness(p)$ is the fitness value corresponding to the Pth parent in the population, D is the delay computed at the intersection based on Webster's formula, d represents the delay experienced in the adjacent intersections and a is the number of cycles during the current traffic simulation. The delay value obtained from the incoming and outgoing intersection is weighted as only a certain portion of the traffic enters the intersection. The weight value computed is based on the assumption of equal distribution of delay among all the phases in the signal.

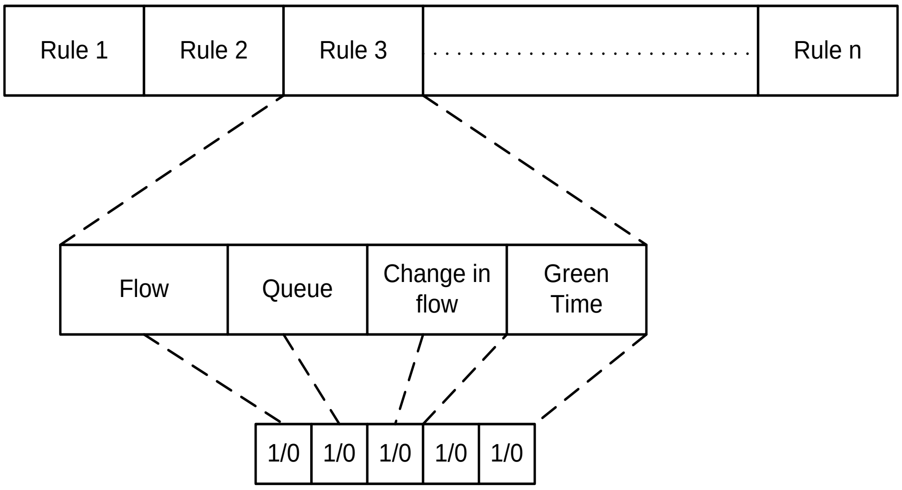


Figure 5.16. Structure of chromosome of the rule base cluster island

The fitness function of the partial solution of the membership function specialization is derived as a function of the fitness value of the parent population and a similarity measure used to compute the similarity between the evolved clusters of an input.

The similarity measure between two different clusters is obtained by constructing a triangle between the mean value and the point of intersection of the line drawn at membership value of 0.1 (Chosen arbitrarily) and the mean of the upper and lower membership function. Euclidean distance measure is used for calculating the distance between the axes points of the triangles constructed. The equations for computing the similarity and distance between two triangles A and B is shown in (5.27) and (5.28).

$$sm(A,B) = \frac{1}{1+DM(A,B)} \qquad (5.27)$$

$$DM(A,B) = \sqrt{\sum_{i=1}^{3}(A_i - B_i)^2} \qquad (5.28)$$

It can be observed that as the distance between the triangles increases, the similarity measure value decreases. Therefore by combining (5.26) and (5.27), the fitness of *l-th* individual in the membership function specialization can be written as shown in (5.29).

$$Fitness_{mf}(l) = \frac{1}{m} * \sum_{i=1}^{n}\big(Fitness(i) * s(i)\big) * \sum_{i=1}^{3}(sm(A_i,B_i) + sm(A_i,C_i) + sm(B_i,C_i)) \qquad (5.29)$$

$$s(i) = \begin{cases} 1 & if\ l \in i \\ 0 & if\ l \notin i \end{cases}$$

Where *m* represents, number of times the specialization was used in the complete solution, A,B,C represents the triangles drawn corresponding to the clusters in the input. Evolution of the GA to minimize this objective function would produce optimal overlap of the cluster as well as a lower travel time delay.

Similarly, the fitness function corresponding to the rule specialization can be written as shown in (5.30).

$$Fitness_{rule}(l) = \frac{p}{m} * \sum_{i=1}^{n}\big(Fitness(i) * s(i)\big) \qquad (5.30)$$

Where P is the number of active rules. When the GA is used to minimize this objective function, number of rules required and the inputs that needs to be active in a rule to produce optimal solution is obtained.

5.6.3. Crossover

The reproduction process does not create new individuals but creates a population from the parent population. New population is created by performing a crossover of two specializations from the same island or cluster. This increases the exploration of the search space and helps in creating new set of parent population. The parents are for the crossover is selected by using binary tournament. One of the parent is selected based on the result of binary tournament. The second parent is selected randomly from the top half of the parents sorted according to the fitness value. One point crossover method was employed to obtain the offspring. The point of crossover is selected randomly.

5.6.4. Mutation

Mutation is an operator whereby the allele of the chromosomes are altered randomly. The real valued part in the membership function specialization cluster is altered by adding a random Gaussian noise with mean zero and variance one. For the binary rule specialization part, the binary value is changed from zero to one or vice versa. Mutation should be used sparingly because it is a random search operator; otherwise, with high mutation rates, the algorithm will become little more than a random search. In our experiment, the mutation probability used was a fixed value of 0.01 .

5.6.5. Reproduction

Reproduction is the process by which the offspring generated are copied to the parent population. The fitness of the evolved offspring is computed and used in the selection of the parent in the next generation. The lower half of the parent for the next generation is selected based on a binary tournament. The top half of the parent for the next generation is obtained as the best half of the parent population obtained by sorting it based on the fitness value. This creates an elitism where the best performing candidate is always retained.

The termination criteria used was the total number of generations. Main reason for such a criteria is each evaluation takes a minimum of 40 seconds and a generation consists of 100 evaluations (10 parents in each cluster). A Single training therefore would require a minimum of 100 hours of simulation.

The evolved final network can be obtained as the combination of the best parent from each of the cluster. This method is based on batch learning as the fitness value at the intersection at the end of the each evaluation is used rather than the fitness value actual time period where there is a variation in the green time of the traffic signal. This is a major limitation as the average value does not truly reflect the traffic condition and can overestimate the performance of solution that performs badly during the peak traffic periods and extremely well during the non-peak hours. Other limitation is the time taken for learning of parameters. This is a serious limitation and cannot be used for real-time learning without high powered multithreaded computing system.

5.7. Q-LEARNING NEURO-TYPE2 FUZZY DECISION SYSTEM

The SET2 decision system developed in the previous section was used to evolve the type-2 fuzzy decision system parameters and its rule base through online batch learning process. The fuzzy system developed decides the optimal green time for a phase in a cycle using the information of flow and queue collected during the phase in progress. The traffic flow and queue value in the other phases were not used for obtaining the green time. This might cause increased delay to be experienced in links that do not have right of way during the current phase in progress. Another major shortcoming with the SET2 decision system is the time required for evolving the type-2 fuzzy network. A better online method that uses the current rewards received from the environment to evaluate and modify the network needs to be developed.

Fuzzy logic systems are generally good at modelling controls based on the rules mined from the expert opinions. However, they lack the ability to learn. The online learning methods that have been developed in the recent past mostly modify only the consequent part of the fuzzy system[118]. The antecedent part of the fuzzy system and their membership functions are usually fixed or designed in an offline manner. In order to perform online learning of all the parameters, a neuro-type2 fuzzy system has been proposed.

The proposed neuro-fuzzy architecture utilises the advantages offered by neural network in learning from data and the capabilities of type-2 fuzzy system. In order to reduce the complexity, Takagi-Sugeno fuzzy sets[119] were employed instead of using Mamdani representation[120]. In simple terms, instead of using a Gaussian functions to represent the consequent part, crisp singleton constant values were used. This greatly improves the mapping of antecedents to the consequents though the interpretation or extraction of rules outside the decision system is difficult.

The general type-2 fuzzy system (Figure 5.1) have a type-reducer before the defuzzification part to obtain a crisp value. The type-reducer converts the type-2 fuzzy set into equivalent type-1 fuzzy set. This is usually performed in an iterative manner or by using geometric methods. Both these methods require the consequents to be sorted arranged in an ascending and descending order. When this type of defuzzification is used in neural networks, a separate memory needs to be used to store the correct mapping (before and after modifying the consequents)[121] and additional overhead is associated with re-mapping the network. In order to avoid this complexity, the type-reducer has been moved before the inference engine in the proposed system. The type-reduction is then performed at the level of the antecedent rather than at the consequents. This would greatly reduce the complexity associated with constructing the neural network.

The type-reduction is performed as the weighted average of the upper and lower firing strength calculated from the membership function for the specific input. This allows the final defuzzification to be performed as a weighted average of the firing strength instead of using centre-of-sets or centroid methods.

Training can be performed in an online manner using Q-learning [122]. Basics of Q-learning techniques were introduced in Chapter 2. In Q-learning, the best state-action pair is evolved over time by actively adjusting the actions chosen for a specific state to maximize the overall scalar reinforcement reward received over time. In conventional learning techniques, the desired output for a specific input state is assumed to be known a priori. The mean squared error value is then computed as the squared difference between the actual output and the desired output. In traffic signal timing optimization problem, the desired output is not known. Therefore adjusting the values needs to be performed using just scalar reward value that does not indicate the desired output but is indicative of performance of the system. Q-learning is useful in such learning problems.

5.7.1. Proposed Neuro-fuzzy decision system

The proposed neuro-type2 fuzzy decision system consists of seven layers as shown in Figure 5.17.

Layer 1 is the input layer that takes in the values of flow, queue and rate of change in flow. Each of the input is n- dimensional, where n refers to the number of phases.

Layer 2 is the fuzzification layer where the inputs are clustered into three regions. Each cluster is modelled using type-2 gaussian function with fixed mean and lower and upper bound sigma values. Therefore each node in layer 2 produces two outputs – Lower and upper membership grade corresponding to the input.

Layer 3 is the type reducer (Figure 5.18), where the type-2 fuzzy input is converted into equivalent type-1 fuzzy set. The synapse between layer 2 and layer 3 assigns the weight corresponding to the lower and upper membership functions. The output of the layer 3 can be represented as in (5.31)

$$\varphi_{ij}^{(3)} = \frac{w_{ljc}^{2} * \varphi_{ijlc}^{(2)} + w_{ujc}^{2} * \varphi_{ijuc}^{(2)}}{w_{ljc}^{2} + w_{ujc}^{2}} \qquad (5.31)$$

Where *c* is the cluster index corresponding to the input, *l* represents the lower and *u* represents the upper bound, *j* represent the input.

Layer 4 represents the rules used to control traffic. Each node in layer 4 represents a rule. The output of the layer is the product of the firing value of the input from each of the input. Since a product T-norm is used in the layer 4, it is essential to perform normalization and is performed in layer 5.

The structure of the rule is significantly different from the rules used in the previous decision systems as it includes inputs for all phases as shown:

"If flow$_{phase1}$ is low and flow$_{phase2}$ is low and.... flow$_{phase\ n}$ is low and Queue$_{phase1}$ is low and .... rate of flow$_{phase\ n}$ is low then green$_{phase1}$ is 0.125 and ...green$_{phase\ n}$ is 0.250"

The output from the layer 5 is the firing value of each rule. However, there is no corresponding consequent assigned as is done in Mamdani system. The weights of the

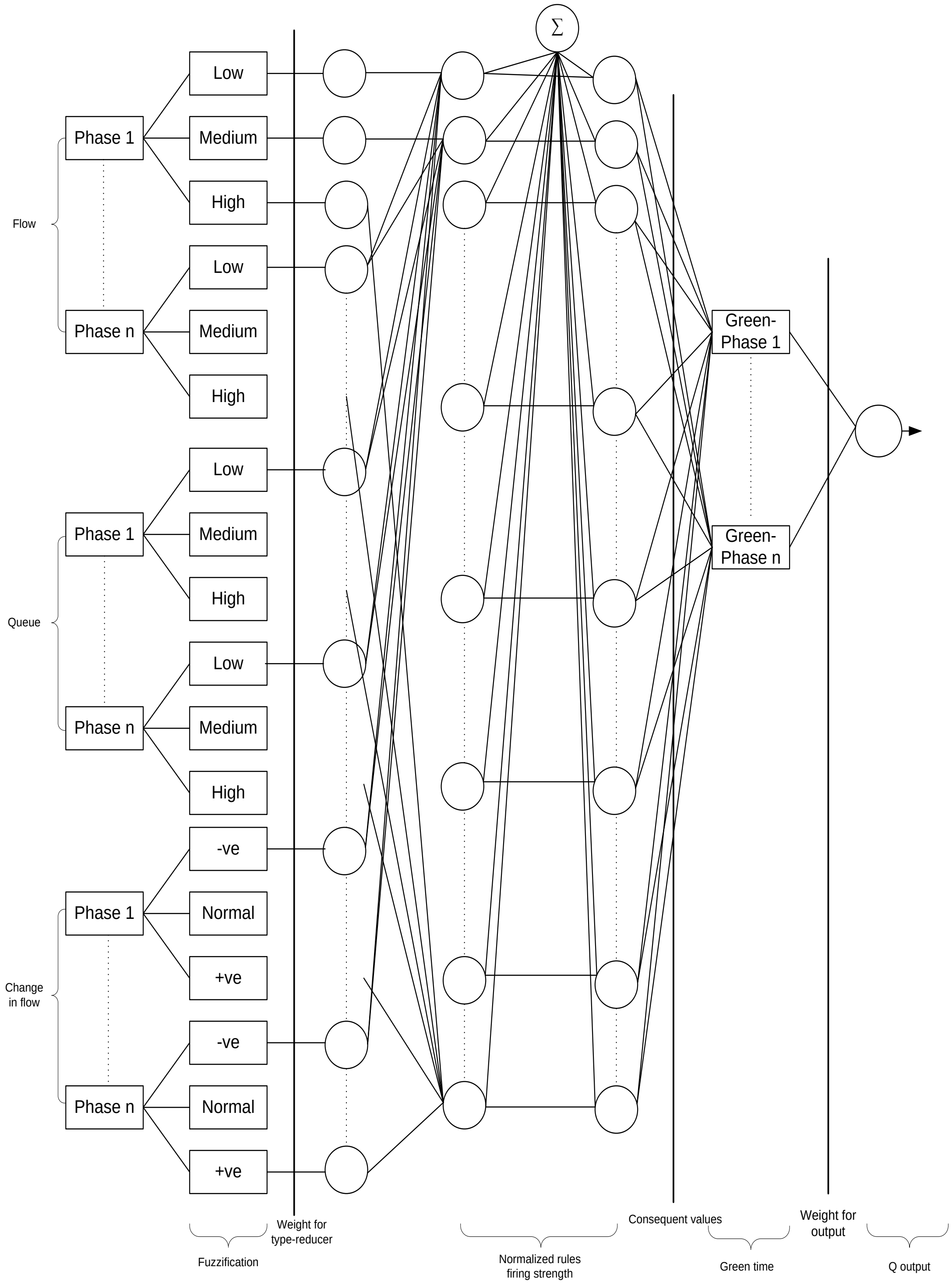


Figure 5.17. Structure of the proposed neuro- type2 fuzzy decision system (QLT2)

synapse connecting the layer 5 with layer 6 represents the consequents. Layer 6 consists of *n* nodes, where *n* corresponds to number of phase. The output of the layer is calculated as weighted average as shown in (5.32).

$$\varphi_i^{(6)} = \frac{\sum_{n=1}^{m} w_{in}^{(5)} * \varphi_n^{(5)}}{\sum_{n=1}^{m} \varphi_n^{(5)}} \tag{5.32}$$

Where *m* is the number of rules and *i* represents the output from layer 6. The final layer consists of single node that provides Q value as the output. The Q-value is represented as in (5.33)

$$\varphi^{(7)} = 40 * \sum_{i=1}^{n} \varphi_i^{(6)} * w_i^{(6)} \tag{5.33}$$

The value of the mean, upper and lower bound sigma values and other weights are adjusted using back propagation of the squared error value. Adaptive learning rate was used during the back propagation as used in [123] The error is calculated using (5.34)

$$E = \frac{1}{2}(TD)^2 \tag{5.34}$$

$$TD = R + \gamma max\ Q(s', a') - Q(s, a) \tag{5.35}$$

Where R is the reinforcement received for taking a specific action in a state, γ is the discount factor which is chosen as 0.1, $Q(s', a')$ is the Q value for the next state and $Q(s, a)$ is the current Q value due to the action chosen in a specific state. Choosing a lower value of discount factor ensures that the update is based on the latest computed Q values. It is much easier to compute the maximum value of the Q if a tabular structure is maintained. However, the state and action space are continuous, it cannot be easily represented using tables and needs to be approximated using generalized models like neural networks. Since the neural network is continuously trained to give maximum value of Q, the max part in (5.35) can be removed and becomes similar to SARSA algorithm. Eligibility trace was not included in this work as value assignment is difficult.

The reward value or the reinforcement received from the road network at each evaluation period is the inverse of sum of total delay experienced at an intersection and discounted value of reward from adjacent intersections.

$$r = {}^{1000}\!/_{R + \sum_{i=1}^{in} R_i * \alpha_i + \sum_{i=1}^{out} R_i * \beta_i} \qquad (5.36)$$

Where R is the reinforcement value at the current intersection, α_i is the ratio of number of phases during which vehicle is released into the network to the number of phases in the incoming agent intersection. β_i is the ratio of number of phases in the current intersection during which vehicles are received by the intersection to the number of phases at the intersection. 1000 is used to make the number large and any positive value could be used here.

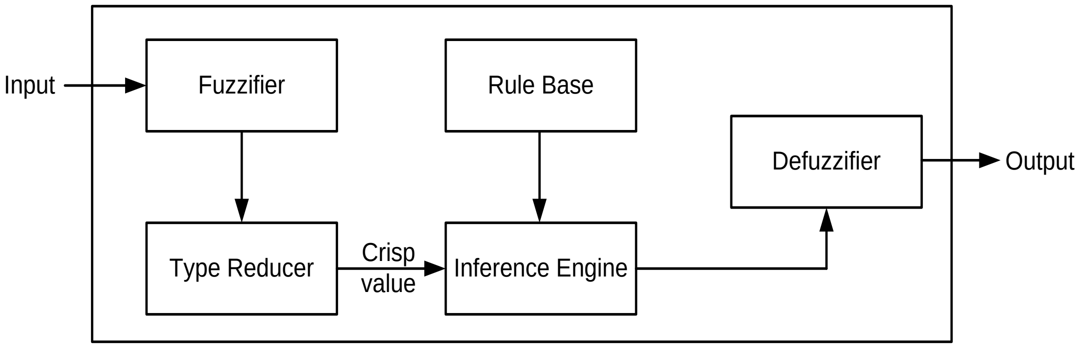


Figure 5.18. Structure of type-2 fuzzy system with modified type reducer

The communication between adjacent agents is an essential component. Without communication of the reward or reinforcement value of the adjacent intersection, the signal green timing computed would cause higher delay. Communication of the Q-values can improve the performance of the system to a great extent, however, it is difficult as the correct state action pair must be known and would require a large memory to store all the values. After the initial fine tuning, it is possible to start the learning after every hour of simulation and keep the network updated.

5.7.2. Advantages of QLT2 decision system

1. Better online learning capability than SET2 decision system.

2. Green time of all the phases are computed at the end of each cycle using a single decision system instead of multiple fuzzy networks or by using a fuzzy network in a sequential manner.

3. Lesser number of evaluations are required than SET2, therefore requiring considerably lesser time to train the network than SET2 decision system.

5.8. SUMMARY

In this chapter, four different types of decision systems were presented. Two of the proposed agent systems were designed based on heuristics and historical traffic volume. The rule base was developed using deductive reasoning technique. T2DR decision system uses a weighted input approach for coordination between agents. The decision system also uses iterative Karnik-Mendel algorithm based defuzzification procedure. GFMAS uses an internal belief model based architecture where the communicated data is an integral part of the decision system. Defuzzification is performed by utilizing the geometric properties of the consequent. The other two decision systems are adaptive in nature with online learning capability. SET2 uses symbiotic evolutionary genetic algorithm to evolve the type-2 fuzzy system parameters and rule base. The fitness function is shared between the neighbouring agents. QLT2 decision system performs online adjustment of weights and parameters of the neuro-fuzzy network using back propagation of the temporal difference error in the computed Q-values. The reward value used to compute the error is obtained as the summation of the discounted reward from the intersections connected to the agent.

CHAPTER 6

SIMULATION PLATFORM

This chapter explains in detail the simulation platform used as a test bed to evaluate the performance of the proposed multi-agent architecture for urban traffic signal control. The simulation platform designed is a large, complicated and realistic problem that can replicate the real-world scenarios efficiently. The chapter details the performance evaluation metrics used to correctly evaluate the performance of the applied control algorithm. The chapter also explains the benchmarks used to compare the proposed multi-agent controllers.

6.1 SIMULATION TEST BED

In order to experiment with different strategies for the application of multi-agent systems for dynamic traffic management and to examine their applicability, a suitably designed test bed is an essential requirement. The main requirements of the test bed developed are the following [124].

- The multi-agent traffic management system should be easily configurable.
- Easy interpretation of the logic developed in traffic terms.
- Ability to create simulated scenarios that best represents the real-world conditions with minimum assumptions
- Provide the necessary complexities associated with urban road intersection, like, the limitations of intersection geometry, vehicle merging etc.

Isolated intersections are commonly used to evaluate the developed traffic signal controls. The main drawbacks of using isolated intersection are as follows:

- The traffic input does not represent the real-traffic flow experienced at urban road networks
- The effect of variation in traffic flow in adjacent network is not captured
- Effects of queue overflow are not represented
- Platoon formation is not efficiently reproduced

These drawbacks critically affect the evaluation of traffic adaptive signal control systems even though they are sufficient in evaluation of smaller traffic networks, where the variation in traffic is limited. This necessitates the use of traffic networks that are interconnected.

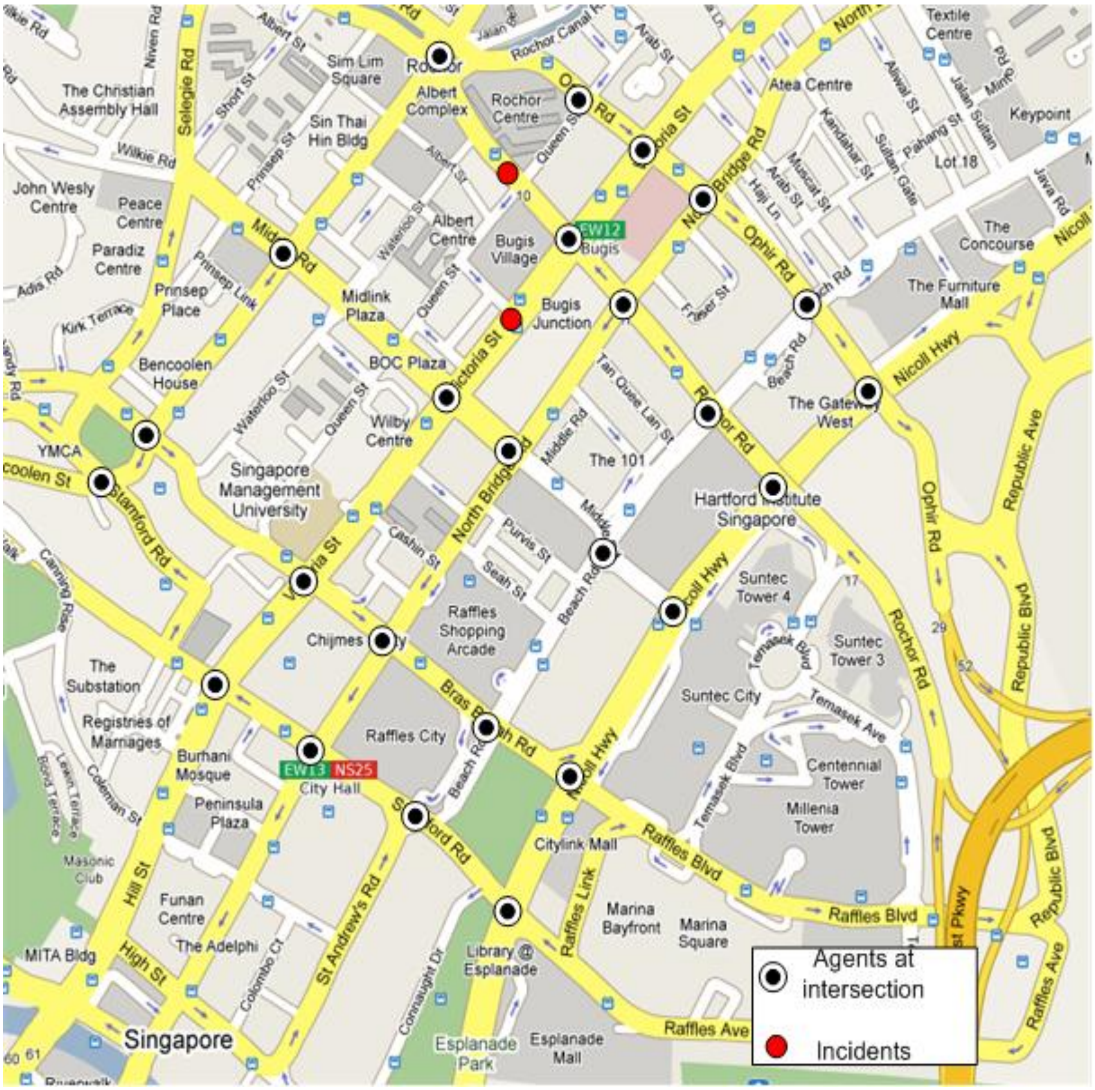


Figure 6.1. Layout of the simulated road network of Central Business District in Singapore

Based on the above requirements, the traffic network selected to evaluate the performance of the proposed multi-agent architecture is a section of the Central Business District (CBD) area of Singapore shown in Figure 6.1. This section of the road network is considered as one of the busiest sections of network that experiences frequent traffic jams due to extreme variation in the traffic flow during the peak periods and different day of the week. The section of the network faces extreme traffic variations due to the presence of large number of commercial offices and shopping centres. This causes the vehicle count to increase considerably during the morning office hours and evening shopping periods well beyond the office timings. This traffic network is significantly bigger compared to those developed in [107, 125-128].

The section of the network consists of one-way links, two-way links, major and minor priority lanes, signalled right and left turning movements, merge lane etc., that makes CBD the perfect test bed to simulate all of the traffic conditions efficiently.

6.2 PARAMICS

The network was modelled using version 6.0 of PARAMICS software. PARAMICS is a microscopic traffic simulation suite developed by Quadstone Ltd [129]. Figure 6.2 Shows a screenshot of the Paramics modeller software. The modeller software is used to define the characteristics of the traffic network, their geometry, amount of traffic to be simulated and the maximum capacity of the network. Paramics allows the traffic process to be simulated on the level of the individual vehicles. The functionality of the Modeller can be further enhanced using customized API. User defined plug-ins , written in C++ and compiled using Microsoft visual C++ compiler allows the users to retrieve traffic simulation information from Paramics and send back the control actions. Critical information like flow, queue and rate of flow can be obtained via the loop detectors in real-time, while the simulation is running. The induction loop detectors are coded in the simulated traffic network at the stop line of each intersection.

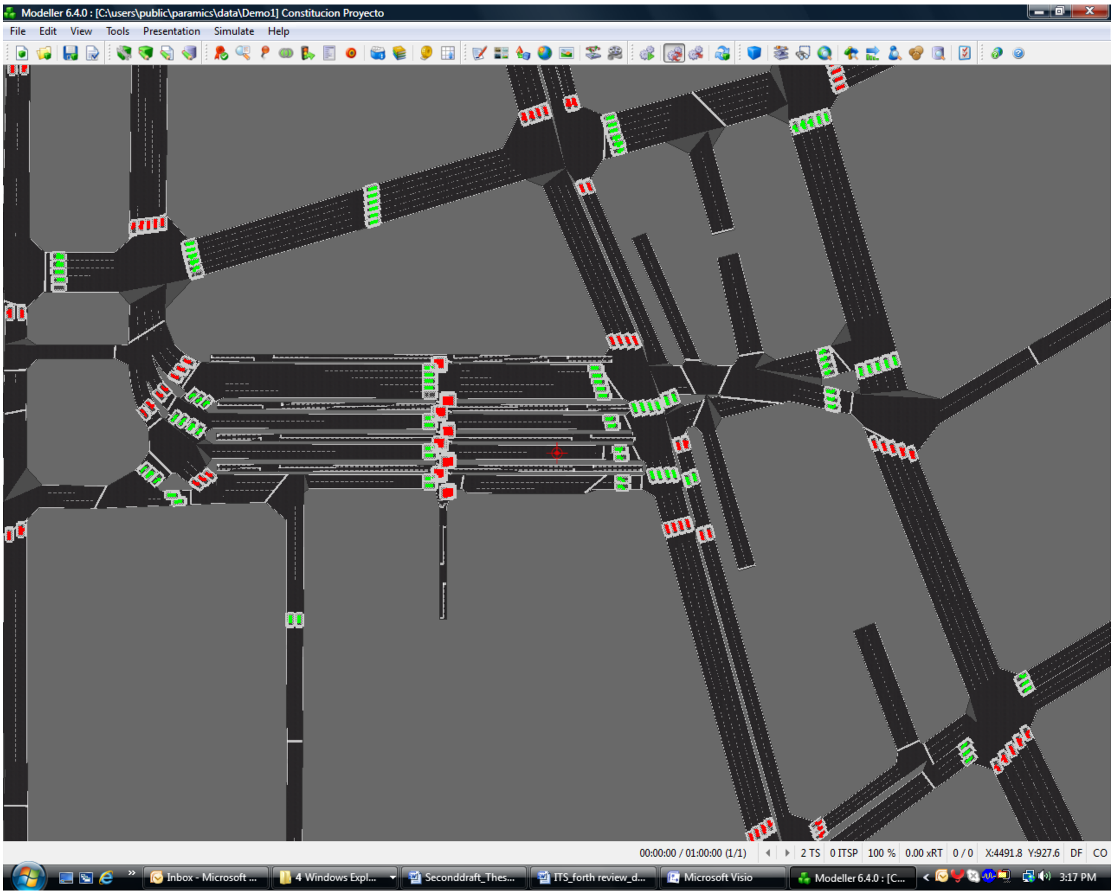


Figure 6.2. Screenshot of Paramics modeller software

6.3 ORIGIN-DESTINATION MATRIX

Details of the simulated traffic network model is as follows:

Total number of nodes	130
Total number of links	330
Total number of zones	23
Total number of loop detectors	132
Total number of vehicle types simulated	16
Total number of agents	25

In order to reflect the true traffic conditions that exist in the urban traffic network of Central Business District of Singapore, the traffic information needs to be collected from the real network and converted into suitable origin-destination matrix. The required data is collected from the Land Transport Authority of Singapore for three separate days. LTA uses GLIDE which is a modified version of SCATS (Explained in Chapter 3). A screenshot of SCATS traffic control is shown in Figure 6.3.

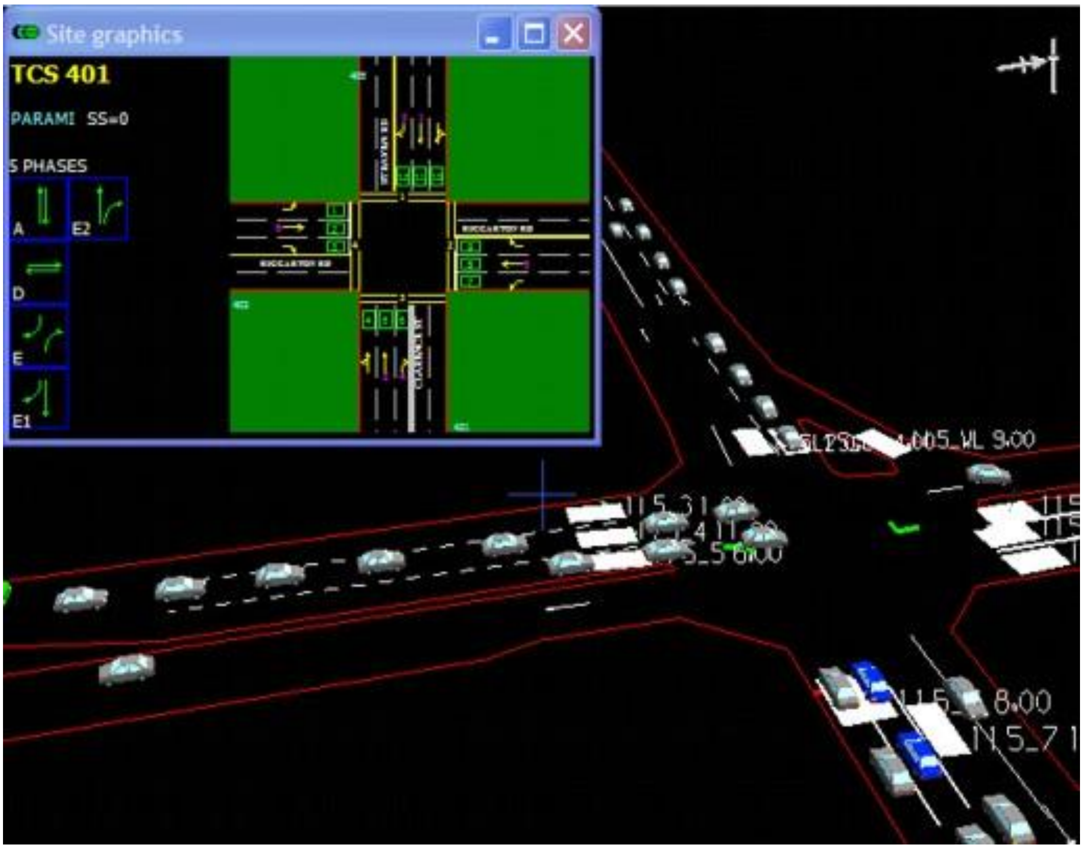
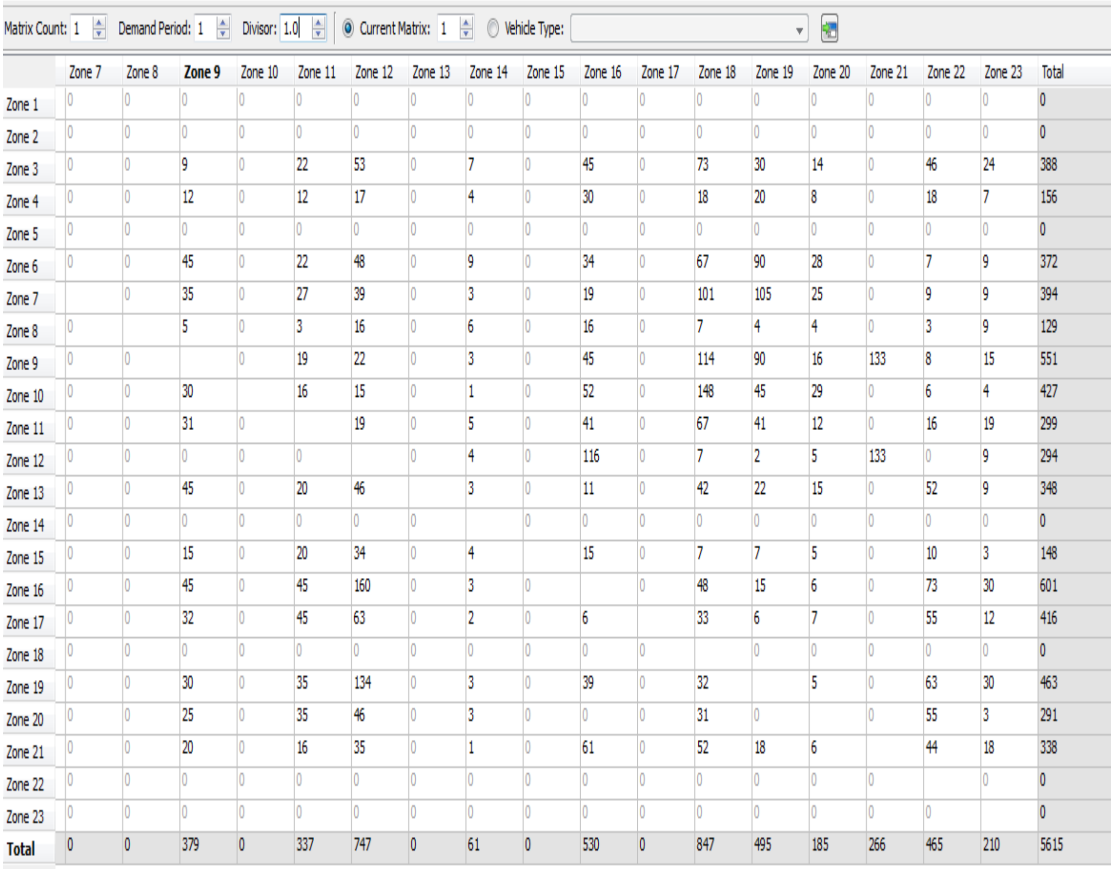


Figure 6.3. Snapshot of SCATS traffic controller and the controlled intersection

SCATS assigns a set of pre-determined signal plans to each intersection according to the traffic condition experienced at the specified time period. Therefore the data collected [108] from LTA consisted of the pre-determined signal plans used for each of the intersection (Computed using the Webster’s signal plan formulation), the order of the signal plan usage during 24-hours over a period of three days, change in the cycle length during the specific signal plan execution and the total number of vehicles that have crossed the specific intersection at each lane during the execution of the specific signal plan. The data is sampled at an interval of 15 minutes which is the usual sampling time for the operation of SCATS. The nominal cycle length is obtained as the average cycle length of the signals during the 15 minute sampling period.



Matrix Count: 1 Demand Period: 1 Divisor: 1.0 ● Current Matrix: 1 ○ Vehicle Type:

	Zone 7	Zone 8	**Zone 9**	Zone 10	Zone 11	Zone 12	Zone 13	Zone 14	Zone 15	Zone 16	Zone 17	Zone 18	Zone 19	Zone 20	Zone 21	Zone 22	Zone 23	Total
Zone 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Zone 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Zone 3	0	0	9	0	22	53	0	7	0	45	0	73	30	14	0	46	24	388
Zone 4	0	0	12	0	12	17	0	4	0	30	0	18	20	8	0	18	7	156
Zone 5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Zone 6	0	0	45	0	22	48	0	9	0	34	0	67	90	28	0	7	9	372
Zone 7		0	35	0	27	39	0	3	0	19	0	101	105	25	0	9	9	394
Zone 8	0		5	0	3	16	0	6	0	16	0	7	4	4	0	3	9	129
Zone 9	0	0		0	19	22	0	3	0	45	0	114	90	16	133	8	15	551
Zone 10	0	0	30		16	15	0	1	0	52	0	148	45	29	0	6	4	427
Zone 11	0	0	31	0		19	0	5	0	41	0	67	41	12	0	16	19	299
Zone 12	0	0	0	0	0		0	4	0	116	0	7	2	5	133	0	9	294
Zone 13	0	0	45	0	20	46		3	0	11	0	42	22	15	0	52	9	348
Zone 14	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
Zone 15	0	0	15	0	20	34	0	4		15	0	7	7	5	0	10	3	148
Zone 16	0	0	45	0	45	160	0	3	0		0	48	15	6	0	73	30	601
Zone 17	0	0	32	0	45	63	0	2	0	6		33	6	7	0	55	12	416
Zone 18	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0
Zone 19	0	0	30	0	35	134	0	3	0	39	0	32		5	0	63	30	463
Zone 20	0	0	25	0	35	46	0	3	0	0	0	31	0		0	55	3	291
Zone 21	0	0	20	0	16	35	0	1	0	61	0	52	18	6		44	18	338
Zone 22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
Zone 23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
Total	0	0	379	0	337	747	0	61	0	530	0	847	495	185	266	465	210	5615

Figure 6.4. Origin-destination matrix indicating trip counts

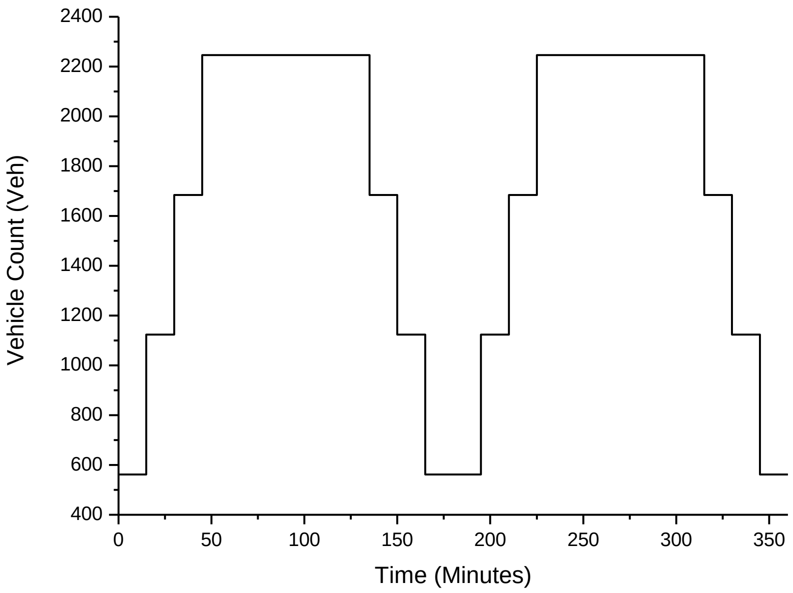
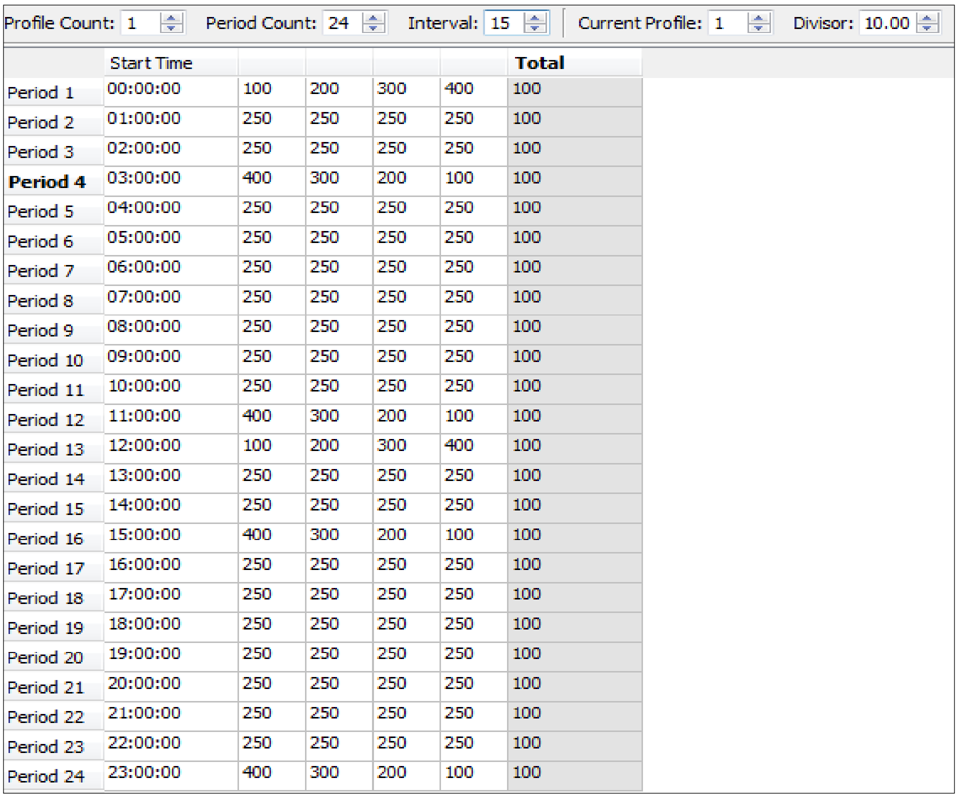


Figure 6.5. Traffic release profile for a six hour, single peak simulation

Using the vehicle count data sampled over a period of three days, a origin-destination matrix is constructed. The road networks usually possess input zones that forms a sink and pool for the release and termination of vehicles. Number of vehicles released from a start zone to the end zone is specified in the demand profile and hourly vehicle release rate is specified in the demand profile which can be adjusted by using the divisor (Figure 6.4).

The profile of the traffic can be adjusted using the divisor value in the profile editor. Figure 6.5 Shows the typical traffic profile for a six hour, two peak simulation and the vehicle release pattern. The modeller software ensures the number of vehicles released into the network to be equal to the percentage of vehicles indicated in the profile (Figure 6.6) from the specified demand matrix. The vehicles are released randomly even though the vehicle count is pre-fixed for the release period (In our case it was fixed at 15 minutes to be consistent with SCATS data sampling period). This property reflects the stochastic nature of traffic witnessed in the real world.



Profile Count: 1 Period Count: 24 Interval: 15 Current Profile: 1 Divisor: 10.00

	Start Time					Total
Period 1	00:00:00	100	200	300	400	100
Period 2	01:00:00	250	250	250	250	100
Period 3	02:00:00	250	250	250	250	100
Period 4	03:00:00	400	300	200	100	100
Period 5	04:00:00	250	250	250	250	100
Period 6	05:00:00	250	250	250	250	100
Period 7	06:00:00	250	250	250	250	100
Period 8	07:00:00	250	250	250	250	100
Period 9	08:00:00	250	250	250	250	100
Period 10	09:00:00	250	250	250	250	100
Period 11	10:00:00	250	250	250	250	100
Period 12	11:00:00	400	300	200	100	100
Period 13	12:00:00	100	200	300	400	100
Period 14	13:00:00	250	250	250	250	100
Period 15	14:00:00	250	250	250	250	100
Period 16	15:00:00	400	300	200	100	100
Period 17	16:00:00	250	250	250	250	100
Period 18	17:00:00	250	250	250	250	100
Period 19	18:00:00	250	250	250	250	100
Period 20	19:00:00	250	250	250	250	100
Period 21	20:00:00	250	250	250	250	100
Period 22	21:00:00	250	250	250	250	100
Period 23	22:00:00	250	250	250	250	100
Period 24	23:00:00	400	300	200	100	100

Figure 6.6. Profile demand editor for a twenty four hour eight peak traffic simulation

6.4 PERFORMANCE METRICS

Performance of the proposed multi-agent signal control is evaluated based on two measures.

1. Mean delay of vehicles

2. Mean speed of vehicles currently inside the network

6.4.1. Travel Time Delay

Delay at each signalized intersection is computed as the difference between the actual travel time of vehicles across the intersection and the travel time in case of no signal control. The actual delay is calculated as the sum of the acceleration, deceleration and stopped time delay for each vehicle at an intersection. In microscopic traffic simulation platform like PARAMICS, the delay value calculated at each intersection of the network at every time step of the simulation is stored in the memory, and an average value of the time delay T_{AD} is calculated as

$$T_{AD} = \frac{\sum_{i=1}^{n} T_D}{T_v} \qquad (6.1)$$

where n is the number of intersections in the road network, T_D is the time delay experienced by vehicles at each intersection and T_v is the total number of vehicles that entered and left the network during the measurement period. Delay parameter has been widely adopted for characterizing traffic signal control schemes. In [130], it has been proved that the average queue size at any intersection is directly proportional to the average delay experienced by a vehicle inside the network, which makes average time delay parameter a suitable entity to classify the congestion level at an intersection. Another important work that showed the linear relationship of the detector occupancy with the average delay was [131]. Moreover, the Highway capacity manual (HCM2000)[132] uses the average control delay incurred by vehicles at the intersection to classify the level of service offered by each signal control. All of these aforementioned works justify suitability of using average delay parameter to evaluate the performance of traffic signal control.

6.4.2. Mean Speed

Current vehicle mean speed is the other parameter used for evaluation of performance of the proposed architecture. It is essential to use the speed parameter along with the delay parameter to avoid errors caused due to inaccuracy in the calculated travel delay value and the number of vehicles inside the network [33, 62]. Current vehicles mean speed is inversely proportional to delay. These two parameters reflect the overall traffic condition in the road network and have been adopted in this work.

$$S_{av} = {\sum_{i=1}^{n} s_i}/{n} \qquad (6.2)$$

Where S_{av} is the average value of vehicle speed inside the network, s_i is the speed of individual vehicles and *n* is the number of vehicles released into the network.

6.5 BENCHMARKS

It is extremely difficult to find a best suitable benchmark for an urban traffic network with a large number of interconnected intersections. Some of the major factors that affect the selection of the benchmarks are as follows:

- Existing traffic signal control algorithms have been specifically developed for different traffic networks with varying complexity.
- The traffic patterns on which the networks have been tested are different from those experienced in Singapore.
- Most of the controllers were tested for isolated intersections.
- Internal working of the commercially available traffic controllers like SCATS, SCOOT and GLIDE are not known and are proprietary in nature.
- The assumptions used are different limiting the application of the developed signal control on the system.

For this specific reason Hierarchical Multi-agent System (HMS) traffic signal control is used as a benchmark in this study. Working of the HMS has been explained in detail in Chapter 3. The HMS have been specifically developed for the Central Business District of Singapore (Same as the network considered in this study) and exactly same traffic scenarios, patterns and software has been used to evaluate the performance. This reduces the requirement of recoding the algorithm that might produce a sub-optimal performance than what was intended.

The other benchmark used is the version of GLIDE simulated in [33, 62]. Though it has been shown that HMS performs better than simulated version of GLIDE in [33], it would be appropriate to show a comparison here as it had been tested on the same traffic network as the proposed signal control and for similar traffic patterns.

6.6 SUMMARY

This chapter describes the various implementation details concerning the modelling of the urban traffic network using PARAMICS software. It details how the data collected from Land Transport Authority of Singapore is converted into origin-destination matrix that reflects the true traffic pattern experienced at the urban traffic network. The chapter also details performance metrics and the benchmarks used to evaluate the performance. The next chapter discusses the simulation results for all the experiments that are carried out in this study to evaluate each of the proposed multi-agent system developed.

CHAPTER 7

RESULTS AND DISCUSSIONS

In this chapter, details of the various experiments that were performed to evaluate the performance of the proposed distributed multi-agent system based traffic signal control developed in this study are detailed. Different types of simulation scenarios were designed consistent with the tests performed in [108] that effectively tests the robustness and effectiveness of the proposed multi-agent systems. Apart from the tests in [108], lane closures and incidents were simulated to test the responsiveness, stability and effect of the variation in capacity of the infrastructure on the proposed multi-agent based traffic signal controls. An in-depth analysis of the simulation results and the reasons for improved performance of the proposed traffic signal controls over benchmark signal controls – HMS (Hierarchical Multi-agent Systems) and GLIDE (Green Link Determine) are presented.

7.1. SIMULATION SCENARIOS

Two types of simulation scenarios were used to evaluate the performance of the proposed multi-agent based traffic control system. They are as follows:

1. Peak traffic scenarios

2. Events

While '*peak traffic scenarios*' simulate the traffic flow pattern observed in urban road networks , '*events*' simulate the traffic scenarios where there is a change in capacity of the road network.

7.1.1. Peak Traffic Scenarios

Three types of peak traffic scenarios were used to evaluate the performance of the multi-agent traffic signal controller. They are namely:

- Six hour, two peak simulation
- Twenty four hour, two peak simulation
- Twenty four hour, eight peak simulation

The six hour, two peak traffic simulation is designed to test the response of the proposed multi-agent traffic signal control to peak traffic conditions experienced during the morning and afternoon period. This test was designed to verify the efficiency of the proposed multi-agent systems to high traffic conditions experienced in a short period of time. Twenty four, two peak simulation reflects the true condition experienced at the Central Business District of Singapore during the morning and evening periods. This scenario creates an infinite horizon problem that replicates the increased stress level experienced by the road networks during peak periods. The final traffic condition experimented is the twenty four hour, eight peak traffic condition. This is an extremely fictitious scenario, where the urban traffic network is subjected to extreme stress condition. Multiple peaks are placed close to each other that creates an increased level of stress condition and allows to test the response of the proposed signal controls to dynamic variation in traffic condition.

7.1.2. Events

Two different type of events were simulated to verify the responsiveness of the proposed multi-agent traffic signal control and the effect of variation in capacity of the road network and are as follows:

- Link and lane closures
- Incidents and accidents

Link and lane closures simulate the conditions where the infrastructure is not available for use due to pre-planned events or conditions. Incidents and accidents, simulate conditions that cause restriction to traffic flow due to reasons not foreseen and accounted for during signal timing optimization.

7.2. SIX HOUR, TWO PEAK TRAFFIC SCENARIO

The two peak simulation is a typical traffic pattern, where the morning and afternoon heavy traffic condition is simulated. The traffic release pattern and the number of vehicles released into the road network during different time periods of simulation is shown in Figure 7.1.

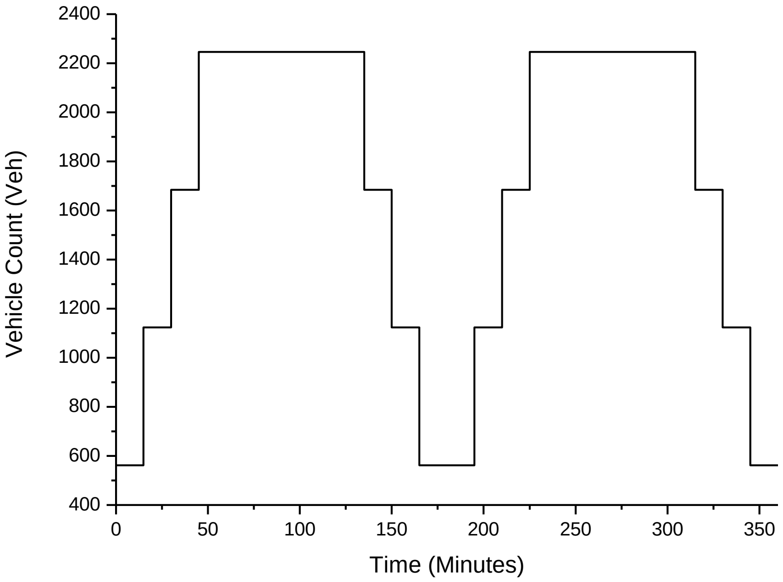


Figure 7.1. Vehicle release profile for a six hour, two peak traffic scenario

Figure 7.1 shows the approximate count of the vehicles released into the network during the period of simulation. The actual traffic release is random and is divided into regions of fifteen minutes period. The number of vehicles released during the fifteen minute period is fixed using the origin-destination matrix.

Table. 7.1. Mean travel time delay and speed of vehicles for a six hour, two peak traffic scenario

Control Techniques	Total mean delay (sec per vehicle)		Current mean speed (kmph)	
	1st peak	2nd peak	1st peak	2nd peak
QLT2	201	209	28	26.5
QLT1	222	218	20.75	23.08
SET2	209	213	23.3	25.16
GAT2	224	221	20.5	21.5
GFMAS	222	219	22.4	14.4
T2DR	224	222	20.32	17.92
HMS	400	470	11.2	12.8
GLIDE	500	600	8	4.8

Table 7.1. shows the comparison of results in terms of travel time delay and mean speed of vehicles inside the road network for the proposed multi-agent signal controls and the benchmarks at the end of peak traffic period. It can be seen that there is a significant improvement in the delay experienced by vehicles during the peak traffic time when using proposed multi-agent based traffic signal controls in comparison to HMS and GLIDE. It can also be seen that for both HMS and GLIDE, the performance degrades during the second peak period and can be attributed to the increased settling time after the end of first peak period.

Among the proposed traffic signal controls, QLT2 performed the best followed by SET2 signal control. GAT2 and QLT1 signal controls had a performance equivalent to those of heuristics based fuzzy signal control. Table 7.1 indicates the point value of delay experienced at the end of the peak traffic period. This value indicates the performance of the signal control during the start of increase in vehicle count. For proper analysis, this value might not be sufficient as the delay is experienced a short period of time after the increase in traffic count. Hence the main reason for the increase in travel time delay needs to be analyzed. Travel time delay is directly proportional to the number of vehicles retained inside the network. Traffic count information at the end of the peak period would not convey much information. Hence an hourly analysis of the number of vehicles released and retained inside the network is performed in Table 7.3.

Table.7.2. Total number of vehicles inside the network at the end of each hour of simulation for six hour, two peak traffic scenario

Simulation time	Number of vehicles inside network								Total number of vehicles released from start	% Vehicles retained						
	GLIDE	HMS	GFMAS	T2DR	QLT2	QLT1	SET2	GAT2		GLIDE	HMS	GFMAS	QLT2	QLT1	SET2	GAT2
0:00:00	28	35	47	17	36	38	36	36	50	56%	70%	94%	72%	76%	72%	72%
1:00:00	694	700	650	416	440	577	440	440	5565	12%	13%	12%	8%	10%	8%	8%
2:00:00	1671	1200	373	878	399	383	469	538	14505	12%	8%	3%	3%	3%	3%	4%
3:00:00	877	750	83	106	95	100	69	79	20120	4%	4%	0%	0%	0%	0%	0%
4:00:00	800	710	622	590	589	539	426	440	25735	3%	3%	2%	2%	2%	2%	2%
5:00:00	1850	1000	398	845	394	377	429	512	34625	5%	3%	1%	1%	1%	1%	1%
6:00:00	900	250	103	101	91	86	93	76	40240	2%	1%	0%	0%	0%	0%	0%

Since the entire traffic flow generation mechanism is stochastic in nature, a comparison between different control strategies is valid only when the number of vehicles released into the road network at the end of specific time periods remains close to each other. This condition is satisfied by using a fixed traffic profile and origin-destination matrix for all simulations. Comparison is done with respect to the best performing benchmark signal control HMS.

The results clearly indicate the effectiveness of proposed multi-agent based signal controls in clearing traffic at intersections, and improving the current mean speed of the vehicles. To prove the repeatability and robustness of the results, ten different simulations were conducted for the scenario with different initial random seeds. The standard deviation of the time delay for the ten simulation runs and the 90, 95 and 99 percentage confidence interval is calculated and is shown in Table 7.3. As can be seen from Table 7.3, even for 99 percentile confidence interval, the delay value fluctuation is restricted to a maximum of ±10.7 seconds. The QLT2 signal controls showed the lowest standard deviation and confidence interval value. T2DR signal controls showed the highest variation. The main reason for this can be attributed to the heuristic nature of the T2DR control and improved learning capability of QLT2. Even though QLT1 controls have similar learning ability, the type-1 fuzzy system have reduces the ability of the signal controls.

Table.7.3. Standard deviation and confidence interval of the mean travel time delay for six hour, two peak traffic scenario

Simulation time		1:00:00	2:00:00	3:00:00	4:00:00	5:00:00	6:00:00
GFMAS	S.D	4.4	10.0	12.0	9.7	11.1	5.9
	90% CI	2.3	5.2	6.2	5.0	5.8	3.1
	95% CI	2.7	6.2	7.4	6.0	6.9	3.7
	99% CI	3.6	8.1	9.7	7.9	9.0	4.8
T2DR	S.D	5.2	8.8	13.1	10.1	12.9	7.0
	90% CI	2.7	4.6	6.8	5.3	6.7	3.6
	95% CI	3.2	5.5	8.1	6.3	8.0	4.3
	99% CI	4.2	7.2	10.7	8.2	10.5	5.7
QLT2	S.D	4.4	5.1	4.9	5.8	4.1	3.2
	90% CI	2.3	2.7	2.5	3.0	2.1	1.7
	95% CI	2.7	3.2	3.0	3.6	2.5	2.0
	99% CI	3.6	4.2	4.0	4.7	3.3	2.6
QLT1	S.D	3.7	4.8	5.9	4.8	5.3	4.2
	90% CI	2.3	5.2	6.2	5.0	5.8	3.1
	95% CI	2.7	6.2	7.4	6.0	6.9	3.7
	99% CI	3.6	8.1	9.7	7.9	9.0	4.8
GAT2	S.D	2.9	4.7	6.0	4.6	3.7	3.4
	90% CI	1.5	2.4	3.1	2.4	1.9	1.8
	95% CI	1.8	2.9	3.7	2.9	2.3	2.1
	99% CI	2.4	3.8	4.9	3.7	3.0	2.8
SET2	S.D	3.1	4.7	4.9	3.4	3.7	2.9
	90% CI	1.6	2.4	2.5	1.8	1.9	1.5
	95% CI	1.9	2.9	3.0	2.1	2.3	1.8
	99% CI	2.5	3.8	4.0	2.8	3.0	2.4

Figure 7.2 shows the average value of the time delay experienced by vehicles for a six hour, two peak traffic profile applied to a traffic network implementing the proposed multi-agent based traffic signal controls. Best results were obtained when traffic signal control was based on multi-agent neuro-type2 fuzzy Q-learning (QLT2) traffic signal controls. This particular behavior can also be observed in the mean speed of vehicles inside the network shown in Figure 7.3. The QLT2 based agent system improved the speed of vehicles during the peak period though all controls performed equally well for low traffic conditions.

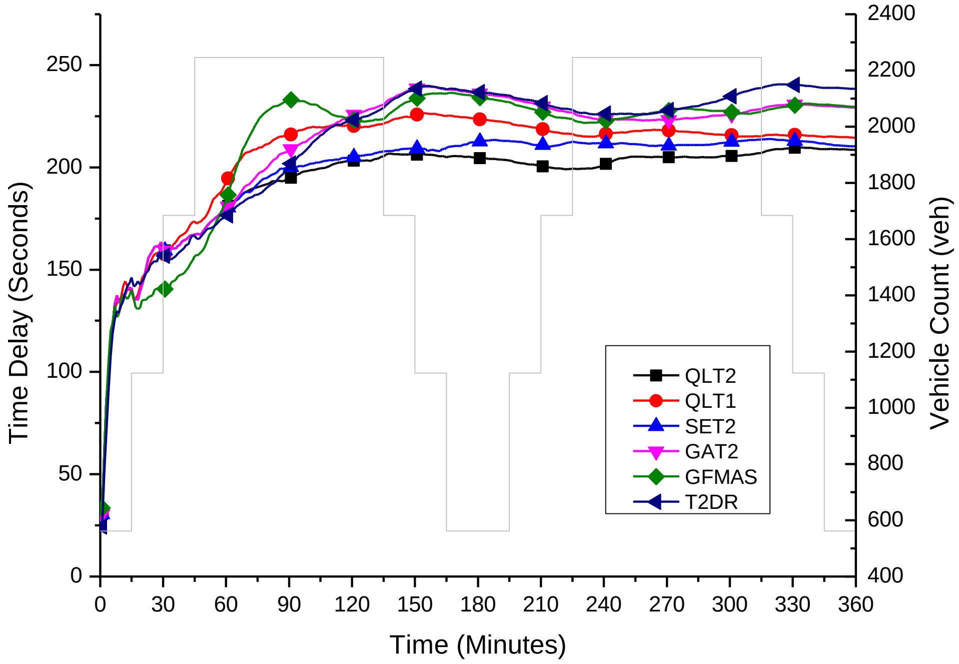


Figure 7.2. Mean travel time delay of vehicles for six hour, two peak traffic scenario

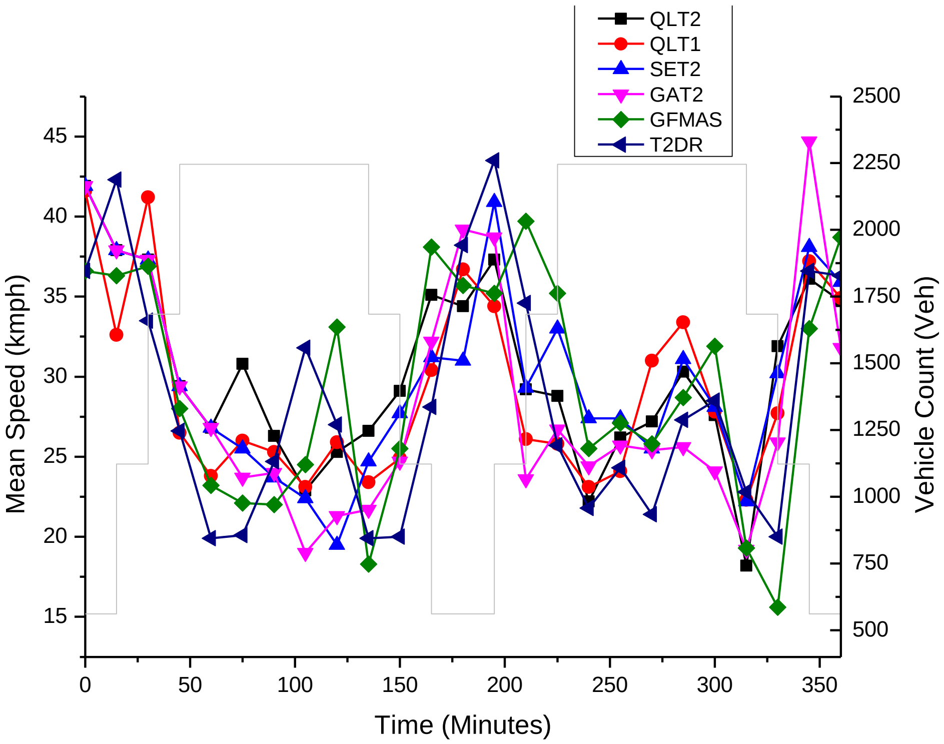


Figure 7.3. Average speed of vehicles inside the network for six hour, two peak traffic scenario

For better understanding, a graphical representation of the number of vehicles retained inside the network during various instances of the simulation is shown in Figure 7.4. It is also necessary to highlight at this point that the speed comparison shown in Figure 7.3. is a sampled version of the real speed data. The actual data fluctuates to a great extent and cannot be presented in a single plot. The actual data is shown in Figure 7.5.

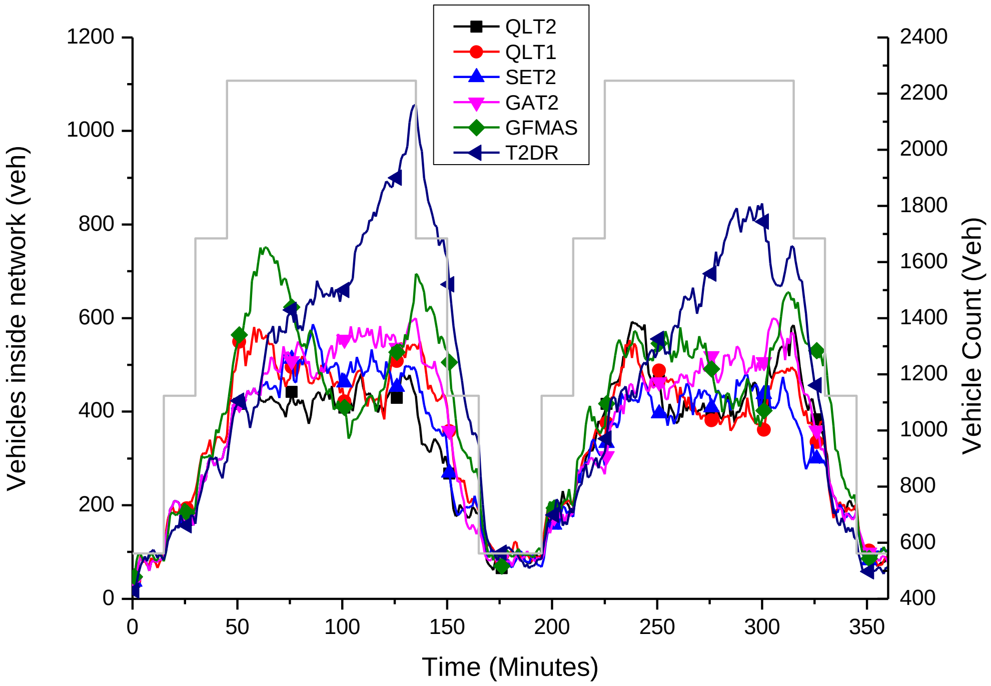


Figure 7.4. Total number of vehicles inside the road network for six hour, two peak traffic

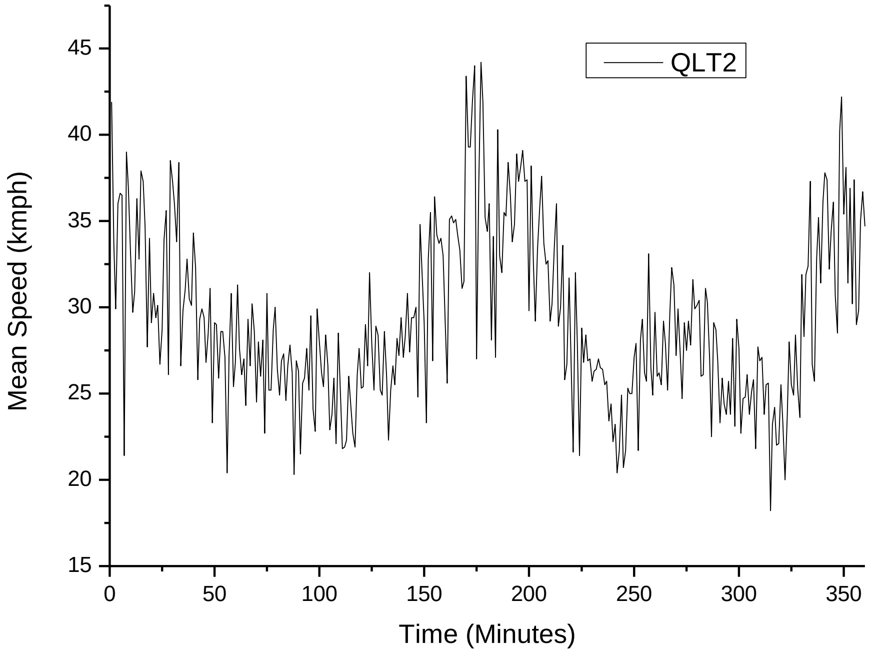


Figure 7.5. Actual mean speed of vehicle inside the road network

Table 7.4. shows the percentage improvement in the observed results over HMS signal control. It can be seen that QLT2 signal controls performed the best followed closely by SET2 signal controls. The GFMAS and T2DR performed better than HMS but the control was not smooth and fluctuated to a large extent as shown in Figure 7.2 and 7.3. The online trained QLT1 signal control performed well, however, the performance was not as comparable to QLT2 control.

Table.7.4.Percentage improvement over HMS signal control

Control Technique	Percentage improvement over HMS	
	Travel time delay	Mean speed
QLT2	52.8	75
QLT1	49.4	52
SET2	51.4	66
GAT2	48.8	58
GFMAS	46	48
T2DR	45.1	46.9

7.3. TWENTY FOUR HOUR, TWO PEAK SCENARIO

For the typical twenty four hour scenario with morning and afternoon peaks, ten simulation runs with different random seeds were conducted for each of the six proposed multi-agent traffic signal controls. The typical traffic release profile is shown in Figure 7.6. The average value obtained in these simulation runs were used to evaluate the performance of the developed signal controllers.

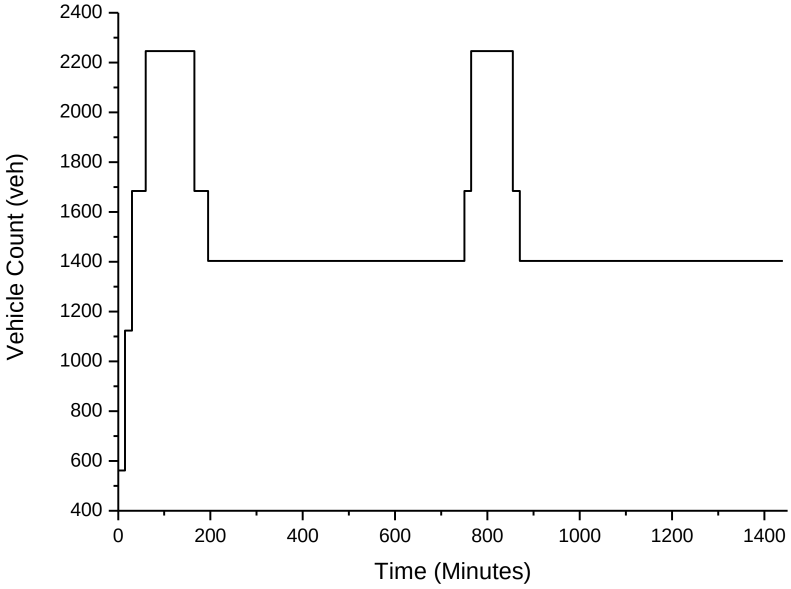


Figure 7.6. Vehicle release traffic profile for twenty four hour, two peak traffic condition

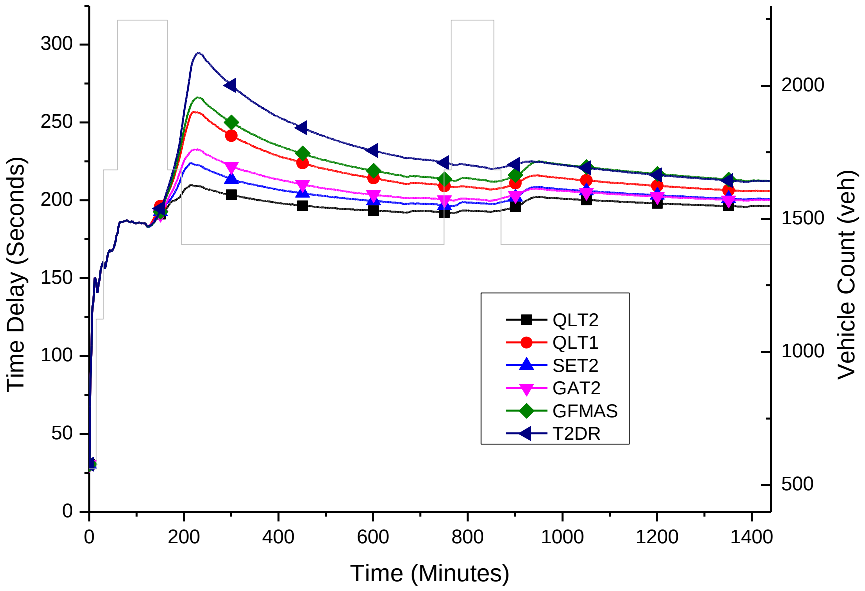


Figure 7.7. Total mean delay of vehicles for twenty four hour, two peak traffic condition

Figure 7.7 shows the comparison of the mean travel time delay experienced by vehicles in response to a twenty four hour, two peak traffic simulation. QLT2 signal control performed the best under this simulation scenario. The second best performance was that of SET2 signal control. However, it was observed that during second peak period, its performance was equal to GAT2 signal control. This observation is supported by the variation in mean speed of vehicles seen in Figure 7.8. The variation in speed clearly indicate a superior performance of QLT2 and low performance of T2DR signal control in comparison to all other proposed multi-agent signal controls.

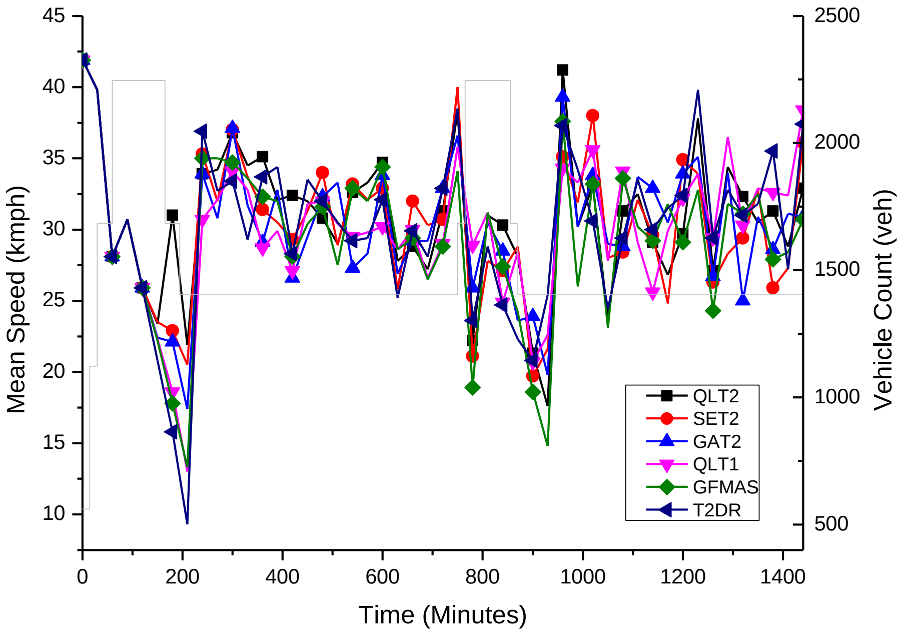


Figure 7.8. Average speed of vehicles inside the network for twenty four hour, two peak traffic scenario

It can be seen from Figure 7.9, there is a considerable amount of fluctuation in the number of vehicles retained inside the road network at the beginning of the second peak period. This particular phenomenon was also observed in the traffic simulation scenario simulated in [108]. This was not seen prominently in HMS as the signal control was not able to reduce the number of vehicles after reaching the peak value and continued to have a flat profile.

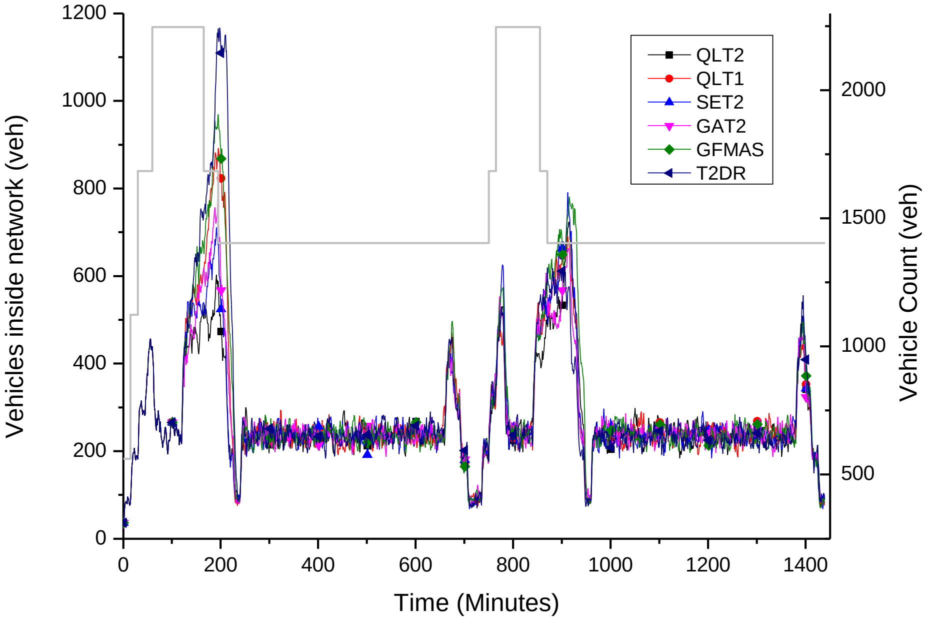


Figure 7.9. Vehicles inside the network for a twenty four hour, two peak traffic simulation scenario

For a more detailed analysis, Table 7.5. shows the comparison of the travel time delay, speed and number of vehicles retained inside the network at the end of the peak periods. QLT2 performs considerably well in comparison to other multi-agent models followed closely by SET2. 47% improvement was observed in the mean time delay and 84% in the speed when using QLT2 over HMS signal control. The heuristically designed T2DR signal control also produced better results than online trained HMS signal control and can be observed in Table 7.6. The main reason for superior performance is due to the fact that the rule base was specifically selected by comparing the delay with HMS signal control. The performance of SET2 signal control degraded during the second peak period. This is mainly because of the traffic pattern used for training the signal control.

SET2 signal control was not able to capture the dynamics of the traffic sufficiently. Further, the green time of each phase was calculated at the end of the phase instead of the end of cycle time. This increases the delay as the vehicles in lanes without right of way are made to wait for longer period. It can also be seen that all proposed controllers performed well without large fluctuations as the change in traffic count is slower in this traffic simulation scenario.

Table 7.5. Comparison of mean delay, speed and number of vehicles for twenty four hour, two peak traffic scenario

Control Technique	Total mean delay (sec per vehicle)		Current mean speed (kmph)		Number of vehicles inside network (Veh)	
	1st peak	2nd peak	1st peak	2nd peak	1st peak	2nd peak
GLIDE	600	430	9.6	8	1800	1650
HMS	420	340	12.8	12.8	1200	1050
QLT2	208.5	193.5	28.1	28.7	418	514
QLT1	250.4	207.8	18.6	24.9	744	533
SET2	222.4	198.7	22.9	27.1	463	538
GAT2	229.5	201.2	22.1	28.5	475	550
GFMAS	256.7	212.5	17.8	24.2	703	549
T2DR	274	221	15.8	22.3	905	619

Table 7.6. Percentage improvement of travel time delay and speed over HMS control for twenty four hour, two peak traffic scenario

Control Technique	Percentage improvement over HMS	
	Travel time delay	Mean speed
QLT2	47.1	84
QLT1	39.7	52
SET2	44.6	71
GAT2	43.3	59
GFMAS	38.4	41
T2DR	35	29

The standard deviation and the confidence interval calculated using ten different random seeds (Table 7.7) was well within the permissible limits indicating a good repeatability and responsiveness to varying traffic loads. T2DR signal control performed the worst in comparison to all other proposed multi-agent controls and QLT2 signal control performed the best with a standard deviation of ±2.9 seconds for the ten simulation runs. It can also be seen that substantial improvement in the standard deviation values and confidence interval was observed during the second peak period. This is mainly due to the ability of the signal control to adapt to repetitive peak traffic conditions.

Table 7.7. Standard deviation and Confidence interval for a twenty four hour, two peak traffic mean travel time delay

		Simulation period	
		1st peak	2nd peak
T2DR	SD	5.1	6.2
	90% CI	2.7	3.2
	95% CI	3.2	3.8
	99% CI	4.2	5.1
GFMAS	SD	4.3	3.2
	90% CI	2.2	1.7
	95% CI	2.7	2.0
	99% CI	3.5	2.6
QLT2	SD	2.9	2.5
	90% CI	1.5	1.3
	95% CI	1.8	1.5
	99% CI	2.4	2.0
QLT1	SD	3.9	5.1
	90% CI	2.0	2.7
	95% CI	2.4	3.2
	99% CI	3.2	4.2
SET2	SD	3.8	2.9
	90% CI	2.0	1.5
	95% CI	2.4	1.8
	99% CI	3.1	2.4
GAT2	SD	4.6	2.1
	90% CI	2.4	1.1
	95% CI	2.9	1.3
	99% CI	3.7	1.7

7.4. TWENTY FOUR HOUR, EIGHT PEAK TRAFFIC SCENARIO

Twenty four hour, eight peak scenario is the extreme traffic pattern simulation stress test to verify the integrity, robustness, and responsiveness of the signal control when subjected to repetitive high traffic condition within a short interval of time. The traffic release pattern is shown in Figure 7.10.

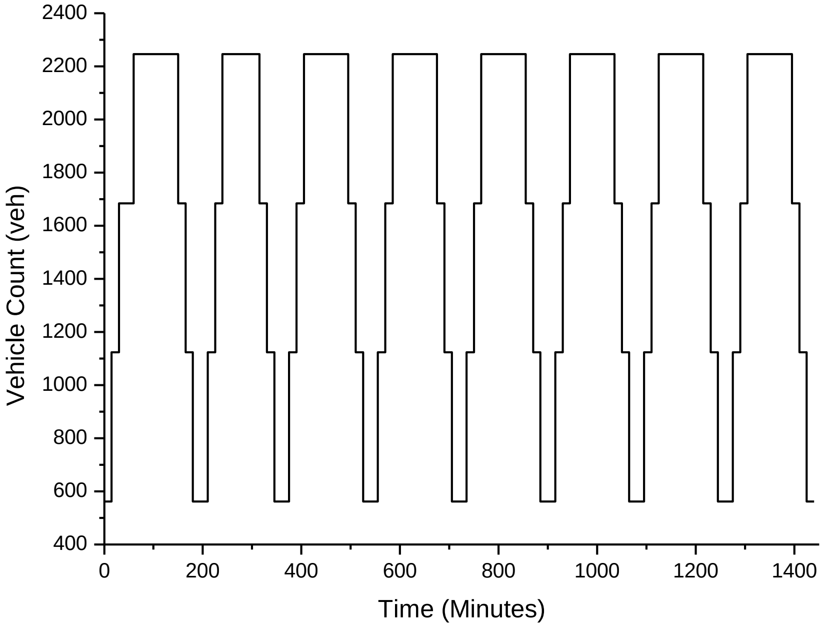


Figure 7.10. Twenty four hour, eight peak traffic release profile

Table 7.8. Travel time delay of vehicles at the end of peak period for twenty four hour, eight peak traffic scenario

control technique	Total mean delay (sec per vehicle) in a period							
	1st peak	2nd peak	3rd peak	4th peak	5th peak	6th peak	7th peak	8th peak
HMS	400	450	450	450	450	500	650	700
GLIDE	400	500	600	650	800	1500	2300	3200
T2DR	242	223	253	243	247	240	247	242
GFMAS	235	243	239	240	242	242	243	240
QLT2	196	207.4	205.8	207.4	206.7	208.8	210.9	210.1
QLT1	203.2	206.9	208.1	210.7	212.2	212.1	216.3	218.2
SET2	204.8	211.5	213.5	212	211	210.8	211.1	212.5
GAT2	195.7	200.3	208.5	213.4	215	216.8	215.4	214.6

It can be seen from Table 7.8 and 7.9, HMS and GLIDE signal control performance starts to degrade after the fifth traffic peak period. The main reason for this degradation is the inability to clear the vehicles present inside the network within a short duration of time before the start of next high traffic period and can be observed from Table 7.10. GLIDE signal control adopts a pre-specified value for the cycle length during the off-peak periods and only changes it to another pre-specified value during the peak period. This makes the green time allocated to each phase insufficient to clear the vehicles when the traffic input increases, resulting in queue spillback. HMS [14][32] signal control also starts showing degradation in performance after the sixth peak period, which is due to the increase in number of vehicles waiting for green time.

Table 7.9. Total mean speed of vehicle inside the network for twenty four hour, eight peak traffic scenario

control technique	Mean speed of vehicles (kmph)							
	1st peak	2nd peak	3rd peak	4th peak	5th peak	6th peak	7th peak	8th peak
HMS	11.2	16	22.4	12	9.6	6.4	6.4	6.4
GLIDE	11.2	8	8	8	0	0	0	0
T2DR	20.8	22.5	18.8	23	11.36	23	12.6	23
GFMAS	21.8	27.2	27.1	23	24.2	20.4	20.6	22.7
QLT2	22.5	28.596	29.496	27	27.696	26.496	26.196	29.904
QLT1	24.996	25.896	27.6	21.696	24.804	30.996	22.404	27.396
GAT2	23.904	29.004	21.204	21.9	25.704	30.3	25.296	26.4
SET2	28.296	29.796	23.7	24.996	27.396	23.004	25.104	28.8

Table 7.10. Vehicles inside the network for twenty four hour, eight peak traffic scenario

		Peak traffic periods							
		1	2	3	4	5	6	7	8
Vehicles inside the network	HMS	2000	1500	1400	1800	2500	3300	3000	3100
	T2DR	700	890	710	819	853	790	793	808
	GFMAS	681	848	701	781	819	754	762	798
	QLT2	427	436	373	441	398	379	509	464
	QLT1	379	401	387	407	330	481	347	386
	SET2	443	466	564	556	507	441	441	394
	GAT2	401	450	444	419	427	460	440	444

The degradation in performance can be attributed to the conflict in decisions between agents of different hierarchy causing smaller green time to be allocated to congested intersections. Therefore the number of vehicles leaving the simulated section of the network increases as the vehicle input is kept a constant based on the OD matrix and can be seen in Table 7.10. However, results show that the proposed GFMAS signal control does not undergo any degradation and performs better than both GLIDE [13] and HMS [19][32]. All of the proposed multi-agent based signal control was able to handle the increased influx of traffic flow without major degradation in their performance.

QLT2 signal control was the best performing multi-agent system in this traffic scenario. Interestingly, SET2 signal control performance started to degrade after the second peak period and was closer to QLT1. On other hand, GAT2 signal control was able to bring down the time delay and improve the speed of vehicles to a greater extent almost equalling the performance of QLT2.

For better understanding of the fluctuations and variations during different simulation runs for the same traffic condition, an analysis of the standard deviation and confidence interval of the observed time delay for every two hours of simulation is presented in Table 7.11. Predictably, GFMAS and T2DR showed large fluctuations in the standard deviation. This indicates that the variation in the observed time delay with different starting points produced varying results.

Table 7.11. Standard deviation and confidence interval of travel time delay for twenty four hour, eight peak traffic simulation

	period	Simulation time (hours)											
		2	4	6	8	10	12	14	16	18	20	22	24
GFMAS	S.D	4.5	8.1	7.6	6.8	5.1	6.4	6.2	6.9	7.3	6.8	6.3	6.0
	90% CI	3.3	6.0	5.6	5.0	3.7	4.7	4.5	5.1	5.4	5.0	4.6	4.4
	95% CI	4.0	7.1	6.7	5.9	4.5	5.6	5.4	6.1	6.4	6.0	5.5	5.3
	99% CI	5.2	9.3	8.8	7.8	5.9	7.4	7.1	8.0	8.4	7.9	7.2	6.9
T2DR	S.D	5.2	9.3	8.3	7.2	6.1	7.2	6.1	6.5	7.6	6.9	6.1	5.8
	90% CI	2.7	4.8	4.3	3.8	3.2	3.8	3.1	3.4	4.0	3.6	3.2	3.0
	95% CI	3.2	5.8	5.1	4.5	3.8	4.5	3.7	4.0	4.7	4.3	3.8	3.6
	99% CI	4.2	7.6	6.8	5.9	5.0	5.9	4.9	5.3	6.2	5.6	5.0	4.7
QLT2	S.D	4.2	4.1	5.3	3.9	2.1	2.6	3.8	3.9	5.1	4.7	4.9	4.9
	90% CI	2.2	2.1	2.8	2.0	1.1	1.4	2.0	2.0	2.7	2.4	2.5	2.5
	95% CI	2.6	2.5	3.3	2.4	1.3	1.6	2.4	2.4	3.2	2.9	3.0	3.0
	99% CI	3.4	3.3	4.3	3.2	1.7	2.1	3.1	3.2	4.2	3.8	4.0	4.0
QLT1	S.D	3.9	2.7	3.9	3.4	3.4	4.1	4.8	5.4	2.9	2.8	4.1	3.7
	90% CI	2.0	1.4	2.0	1.8	1.8	2.1	2.5	2.8	1.5	1.5	2.1	1.9
	95% CI	2.4	1.7	2.4	2.1	2.1	2.5	3.0	3.3	1.8	1.7	2.5	2.3
	99% CI	3.2	2.2	3.2	2.8	2.8	3.3	3.9	4.4	2.4	2.3	3.3	3.0
SET2	S.D	2.4	5.2	4.5	3.8	3.3	4.1	2.7	4.3	4.8	4.4	5.1	2.2
	90% CI	1.2	2.7	2.3	2.0	1.7	2.1	1.4	2.2	2.5	2.3	2.7	1.1
	95% CI	1.5	3.2	2.8	2.4	2.0	2.5	1.7	2.7	3.0	2.7	3.2	1.4
	99% CI	2.0	4.2	3.7	3.1	2.7	3.3	2.2	3.5	3.9	3.6	4.2	1.8
GAT2	S.D	3.1	4.9	5.5	5.2	4.8	3.9	3.3	3.3	4.7	4.2	4.2	3.8
	90% CI	1.6	2.5	2.9	2.7	2.5	2.0	1.7	1.7	2.4	2.2	2.2	2.0
	95% CI	1.9	3.0	3.4	3.2	3.0	2.4	2.0	2.0	2.9	2.6	2.6	2.4
	99% CI	2.5	4.0	4.5	4.2	3.9	3.2	2.7	2.7	3.8	3.4	3.4	3.1

On the whole, a 59% improvement in the time delay and 71% improvement in the speed of vehicles were observed for QLT2 signal controls over HMS. Least improvement was observed in T2DR controls. It is interesting to note that GAT2 signal control performed better than SET2 signal control. Table 7.12 shows the percentage improvement in reducing the travel time delay and increasing mean speed of vehicles inside the network over HMS signal control.

Table 7.12. Percentage improvement of travel time delay and mean speed over HMS signal control

Control Technique	Percentage improvement over HMS	
	Travel time delay	Mean speed
QLT2	59.2	71
QLT1	57.3	66
SET2	54.4	62
GAT2	58.2	70
GFMAS	51	59
T2DR	49.9	57

Figure 7.11 and 7.12 shows the average value of time delay and speed experienced during simulation runs with different random seeds respectively. A distinct lag between the travel time delay and the improvement in speed can be observed. Further, QLT2 and GAT2 shows a low level of fluctuation in mean speed of vehicles indicating a smooth and consistent control. T2DR shows the largest fluctuation and can be deduced as the least performing signal control among the proposed multi-agent systems.

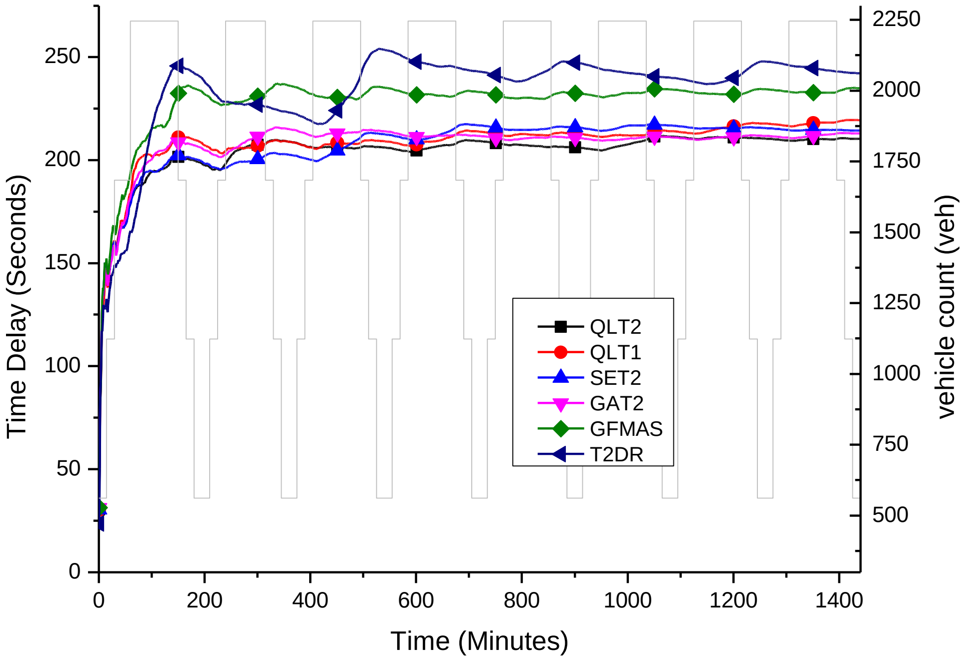


Figure 7.11. Total mean delay experienced for a twenty four hour, eight peak traffic scenario

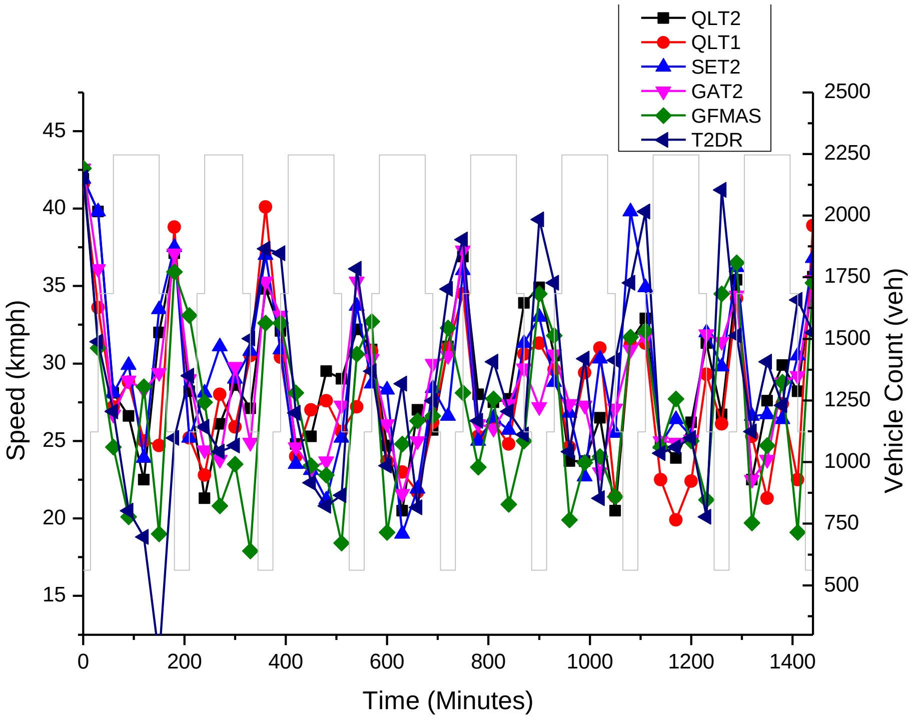


Figure 7.12. Mean speed of vehicles for twenty four hour, eight peak traffic scenario

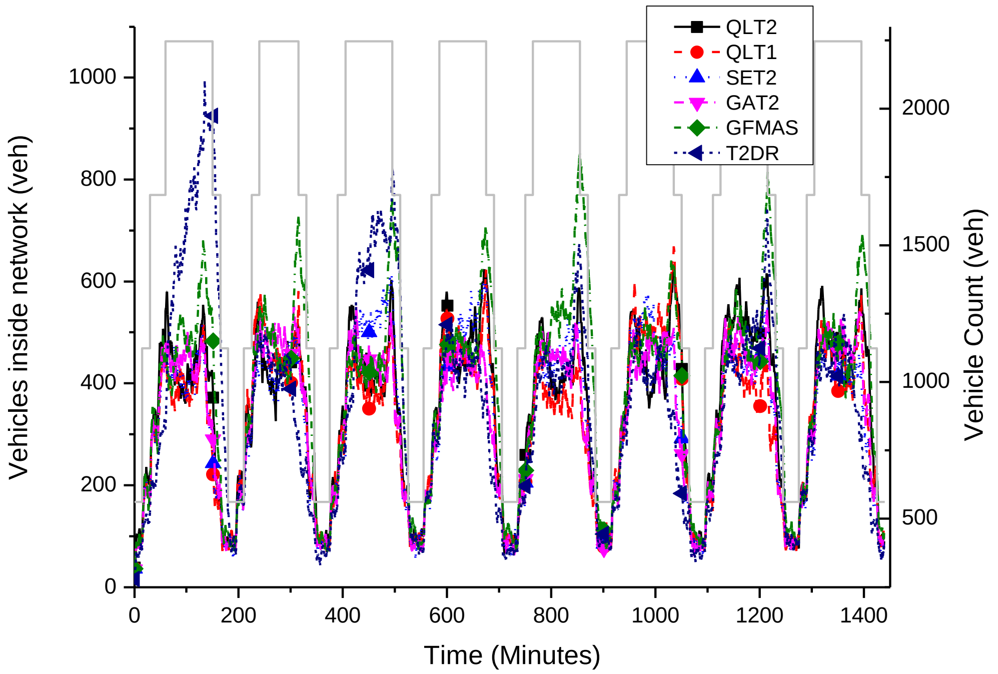


Figure 7.13. Number of vehicles inside the network for twenty four hour, eight peak traffic scenario

7.5. Link and lane closures

Link and lane closures are of practical importance and recreate the conditions of road blockage due to pre organized or planned events such as clearance of roadside trees, scheduled maintenance of traffic management systems like VMS (Variable message signs), ERP (Electronic road pricing units) maintenance, or special events like "Formula One car racing". Planned events reduces the capacity of the traffic infrastructure. Total traffic handling capacity of the road network was reduced by closing down lanes in the road structure. This scenario is similar to step input in control system.

The average travel time delay of vehicles when experiencing multiple peak traffic scenario with two major lane closure is shown in Figure 7.14. There is a shift in the total travel time delay and a increase in the delay during the start period. The main reason for this is the routing mechanism used in PARAMICS. The routing is usually performed by calculating the feedback costs associated with each link network wide for every pre-specified interval (five minutes in this case). This causes the vehicles to opt for major links thereby achieving better optimization and lesser travel delay time. This particular effect is not prominently witnessed when only a single lane is closed. This behaviour can be seen in Figure 7.15. This experimentation proves the reliable performance of the proposed signal controls to the change in infrastructure capacity. All the plotted results are the average of ten simulation runs.

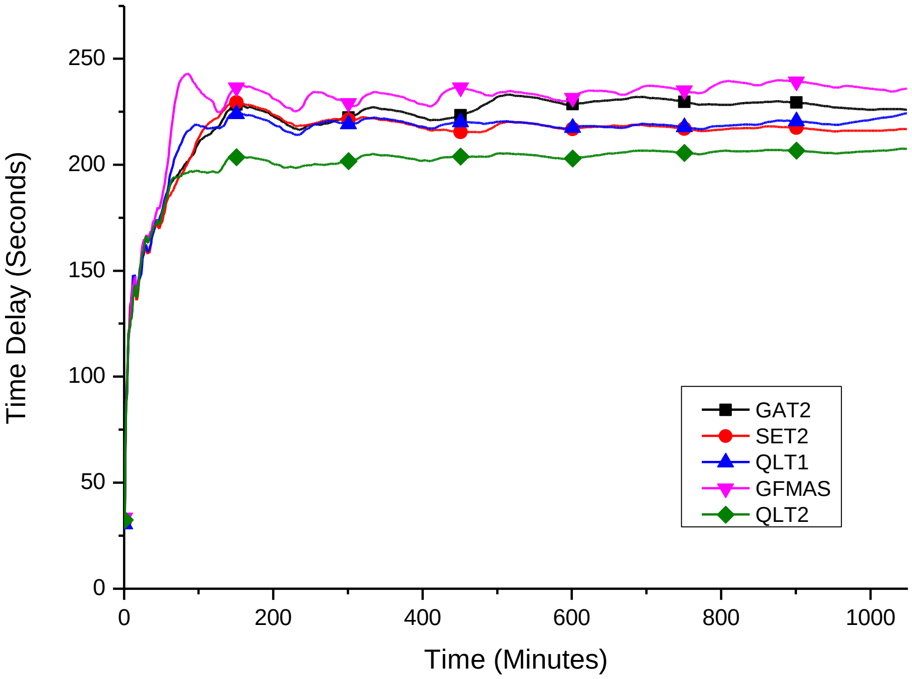


Figure 7.14. Two lane closure – Mean travel time delay of vehicles

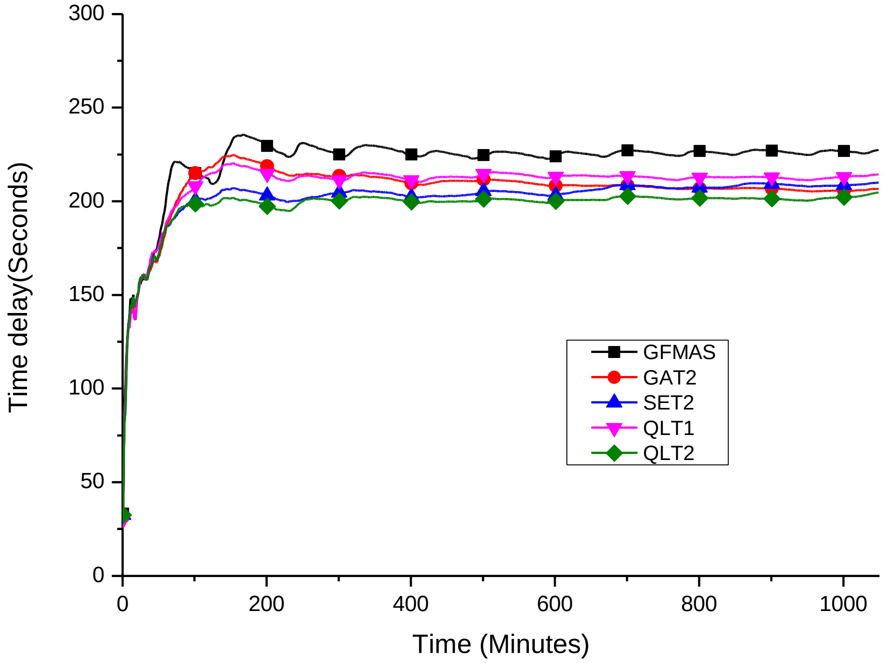


Figure 7.15. Single lane closure – Mean travel time delay of vehicles

7.6. INCIDENTS AND ACCIDENTS

Incidents and accidents simulation try to replicate the random reduction in the infrastructure capacity due to unforeseen reasons like vehicle crash, that cannot be controlled. Incidents were created randomly during the first peak period of the traffic simulation using the incidents file in PARAMICS. Responsiveness of the proposed multi-agent traffic signal controls to such intermittent disturbances, which are analogous to impulse inputs, were studied using this scenario. The detection and clearance of the incidents is assumed to be handled by different sub-units of the traffic management system which do not come under the multi-agent signal control architecture currently.

Figure 7.16-7.17 shows the various scenarios simulated for this study. The incidents were created during the first peak period to examine the response and time taken to settle down. The incidents were simulated in the link connecting Victoria St with Rochor Road, and in the link connecting Rochor Road to Bencoolean St. as indicated in the network map in chapter 6. It can be observed that two simultaneous incidents at different links cause the proposed multi-agent traffic signal control performance to degrade. Although the incidents were simulated very close to the peak traffic period, the increase in the traffic is considerably lesser and the proposed multi-agent traffic signal controls were able to effectively handle the increased traffic congestion, and bring down the average delay experienced by the vehicles.

GFMAS experienced the largest fluctuation in travel time delay. QLT2 signal control showed the least variation in the travel time delay. The behaviour of all signal controls were on expected lines as seen in the twenty four hour eight peak traffic simulation scenario. The only difference was the increased traffic experienced at the start period due to incident. The test conclusively proves the ability of the proposed multi-agent traffic signal controls in handling unexpected change in the traffic pattern.

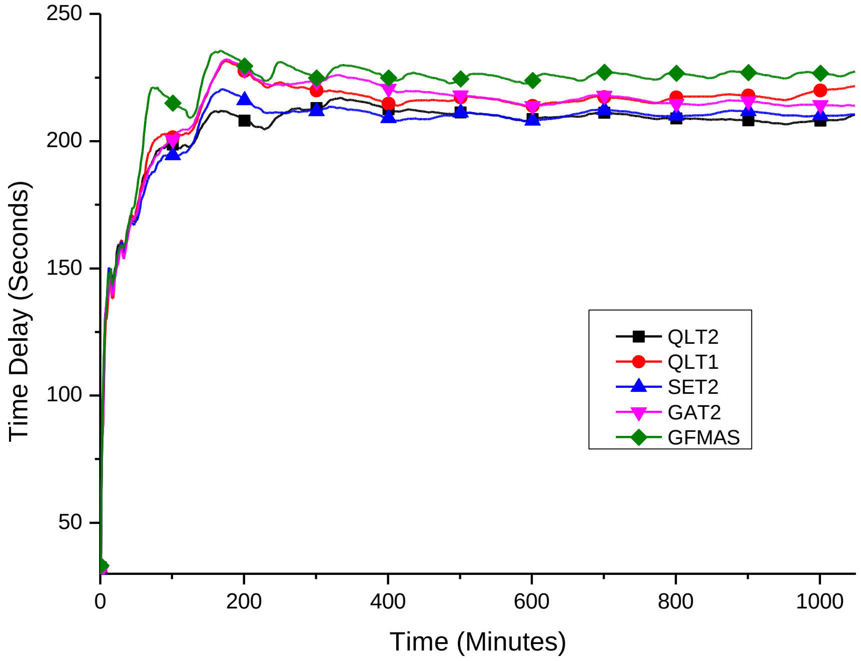


Figure 7.16. Single incident simulation – Multiple peak traffic scenario

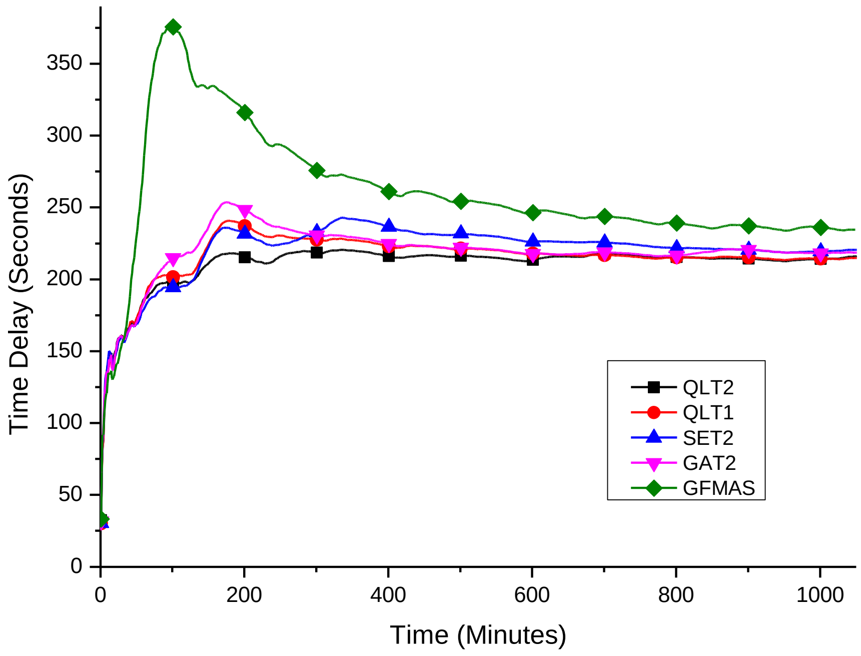


Figure 7.17. Two incidents simulation – Multiple peak traffic scenario

In conclusion, QLT2 signal control performed the best under all traffic simulation scenarios, both for the regular traffic simulation scenarios and events. The performance could be attributed to the online learning feature of the QLT2 signal control. Though QLT1 signal controls also have online learning capabilities, their performance were not equivalent to QLT2 because of the absence of type-2 fuzzy decision system.

Table 7.13. Comparison of the proposed signal control methods with HMS in terms of computation and communication

Control Method	Computational Complexity	Computational Cost	Communication Overhead
QLT2	Low	Low	Low
SET2	High	Low	Low
GFMAS	High	High	Low
T2DR	High	High	Low

The batch learning multi-agent systems SET2 and GAT2 did not perform well because the learning was based on the average fitness function values computed over a period of three hours. This dilutes the difference in fitness function between various multi-agent signal controls and reduces the ability of the system to handle the variation in traffic during lower peak period. This increases the total baseline travel time delay experienced by the vehicles and an overall shift in the value. SET2 performed better than GAT2 because of the increased coverage of state action space than GA. The evolution of membership functions parameters and rule base as two different individual entities in SET2 ensures better exploration of state space. Since the fitness function for the individual population is shared, it allows co-evolution of the parameters. This is the main reason for the improved performance over HMS.

HMS signal control used an actor-critic based reinforcement learning. Actor-critic network are sensitive to exploration and requires the selected actions to be randomly perturbed to improve performance. However, the developed system did not have any such feature to improve exploration. The second disadvantage is the policy selected must be kept fixed for proper learning. However, the rule base was evolved using genetic algorithm removing the reference point for critic learning. The back propagation method used in HMS selectively updated only active rules based on a threshold value. Improper selection of threshold prevents weight values and parameters to be updated. These are few of the reasons why the proposed methods were able to perform better than the HMS signal control method. Table 7.13 shows a comparison of the performance of the proposed traffic signal controls over HMS signal control in terms of computational complexity, computational cost and communication overhead. QLT2 signal control has the lowest in all of these parameters as the control uses a single iteration level each time to decide the actions and learning whereas in SET2 separate memory is required to store the fitness values during the entire period of training pattern inputs.

7.7. SUMMARY

In this chapter, the various traffic simulation scenarios as well the results obtained have been presented and analyzed in detail. A comparative analysis of the results show the better performance of the proposed multi-agent traffic signal control methods over HMS traffic signal control. The reasons for such a marked improved have been presented and discussed. From the results, it was observed that the online learning based systems outperform heuristically designed controls. Among the learning systems, QLT2 performed the best. The following chapter will draw a conclusion to this dissertation and explore the open research avenues for future research work.

CHAPTER 8

CONCLUSIONS

This chapter concludes the dissertation and provides recommendations for the future research that could enhance the functionality of the proposed multi-agent system.

8.1. OVERALL CONCLUSION

The objective of this thesis was to develop a distributed, multi-agent based approach to traffic signal timing optimization and control. The choice for a distributed approach was motivated by the fact that a centralized traffic control approach is often not feasible due to computational complexity, communication overhead, and lack of scalability.

The creation of a distributed, multi agent approach requires the subdivision of the traffic control problem into several loosely coupled sub-problems, such that the combination of all the solutions of the sub problems together provide an approximate solution to the original traffic control problem.

In the multi-agent framework proposed in this dissertation, each agent located at the intersection tries to optimize the green timing of the intersection rather than the whole network with an objective to minimize the travel time delay and increasing mean speed of vehicles inside the road network. In order to perform this, four computational intelligent decision systems have been proposed. Type-2 fuzzy sets was used as the main component of the intelligent decision system. The ability of type-2 fuzzy sets to handle the level of uncertainties associated with the data and stochasticity associated with the dynamic environment, it is an ideal candidate for use in traffic signal timing optimization.

Two of the proposed decision system (T2DR and GFMAS) were designed based on heuristics and the rule base for the type-2 fuzzy sets were obtained by deductive reasoning. This approach performed reasonably well during the high traffic conditions, however, the performance degraded when subjected to a high stress traffic condition.

Third proposed decision system (SET2) exhibits better adaptation than those designed using heuristic methods. It used online batch learning method to adapt the parameters of the type-2 fuzzy sets and at the same time evolve the fuzzy rules. Stochastic optimization technique using symbiotic evolutionary genetic algorithm was able to evolve the parameters better than the traditional GA approach. The cooperative co-evolutionary approach based on fitness sharing between clusters and the neighbouring agents was able to provide better results compared to GA with fitness sharing.

The last proposed decision system was an online learning neuro-type2 fuzzy system whose parameters were adapted every evaluation period unlike the SET2, where the parameters were updated after the completion of a simulation run. The update is based on the objective to maximize the overall reward received by an agent using back propagation technique. The method also combined decision system for all the phases into a single network unlike the other three approaches. This considerably improved the performance over all other proposed multi agent systems and the benchmark multi-agent system.

8.2. MAIN CONTRIBUTIONS

The main contributions of this research were in the conceptualization, development and application of a distributed multi-agent architecture to urban traffic signal timing optimization problem. The significant contributions made in the design front are as follows.

- The development of a generalized distributed multi-agent framework with hybrid computational intelligent decision making capabilities for homogeneous agent structure. The modular concept used in the design allows the reuse of components without major modifications to its internal structure.

- The development of deductive reasoning method for the construction of membership functions, rule base of type-2 fuzzy sets and calculating the level of cooperation required between agents. Manual clustering of the data and fine tuning of the rule base created using expert knowledge through trial and error method to achieve lower travel time delay and improved mean speed of vehicles inside the road network.

- The development of cooperation strategies in multi-agent system through internal belief model by incorporating communicated neighbour agent status information. Two different structures with communicated neighbour status data as an integral part of decision system and as an auxiliary input external to the decision system were experimented.

- The development of symbiotic evolutionary learning method for coevolving membership functions and rule base for the type-2 fuzzy decision system. Modified the general symbiotic evolutionary method to coevolve the cluster mean and spread along with the number of rules and significant inputs in each rule. Comparison with genetic algorithm based evolution showed an improved performance while using modified symbiotic evolutionary learning for evolving parameters of type-2 fuzzy sets.

- The development of modified Q-learning technique with shared reward values for solving distributed urban traffic signal control problem. Adapted the general Q-learning method to a distributed problem by sharing the reward values to improve the global view and prevent premature convergence.

- The development and relocation of the modified type-reducer using neural networks to reduce the computational complexity associated with sorting and defuzzification process in interval type-2 fuzzy sets.

- The development of traffic simulation scenarios to test the reliability and responsiveness of the developed traffic signal controls.

8.3. RECOMMENDATIONS FOR FUTURE RESEARCH WORK

Considerable amount of work has been done by researchers in the area of multi agent systems application to traffic control. However, a solid multi agent framework with hybrid computational intelligent techniques haven’t been developed. Most of the systems developed exhibits only partial or weak agency. Further, the field of multi agent system by itself is a relatively new field with a lot of open avenues for research. Some of the recommendations for future research work are given below.

- The proposed multi agent architecture was designed specifically for the urban traffic signal control problem. However, there are many other applications that are similar to traffic control problem and have similar restrictions. Network packet routing, ATM networks are examples of such similar systems. In order to effectively use the proposed multi agent system for such application, it is essential to generalize the framework and create standard templates that can be easily embedded into the custom codes.

- In this dissertation, the offset timing and direction of coordination were kept fixed. The main reason is the non-availability of the network wide performance information. For improving the performance further, a distributed method to obtain the offset value must be developed. In HMS, the offset adjustment was possible because of hierarchical nature of the system and regional control agents had a better view of a section of the network.

- In the proposed multi agent architecture, the protocol used was similar to FIPA protocol but not all the functionalities were included. For example, service request and acknowledgement were not used as the agents were homogeneous and had the same functionality with no delegation of duty to adjacent agents. However, to connect to legacy systems used in traffic signal control all the functionalities needs to be introduced.

- Parallel evaluation of multiple solution of an agent must be developed using multithreading feature. In the current architecture, the multithreading or parallelization is at the level of agent and not used in the internal evaluation. This is essential to test multi agent system for applications with rapid changing environment.

- The Q-learning approach implemented in our study communicated or passed reinforcement or reward values among the agents. This is a scalar quantity and provides very little direction towards optimal solution. Communicating the value function or Q-values would improve the performance to a great extent. However, the challenge is in storing the state action pair values for the continuous input and perform update in a distributed manner.

LIST OF PUBLICATIONS

JOURNALS

1. Balaji P.G and D. Srinivasan, "Type-2 fuzzy logic based urban traffic management," in *Engineering Applications of Artificial Intelligence journal*,vol.24, no.1, 2011.
2. Balaji P.G and D. Srinivasan, "Distributed Geometric Fuzzy Multi-agent Urban Traffic Signal Control," in *IEEE Transactions on Intelligent Transportation Systems*, vol.11, no.3, pp.714-727, 2010.
3. Balaji P.G, X. German and D. Srinivasan, "Urban Traffic Signal Control Using Reinforcement Learning Agents," in *IET Intelligent Transport Systems*, vol.4, no.3, pp.177-188, 2010.
4. D. Srinivasan, C.W. Chan and Balaji P.G, "Computational intelligence-based congestion prediction for a dynamic urban street network," in *Neurocomputing*, vol.72, no.10-12, pp. 2710-2716, 2009.
5. Balaji P.G and D.Srinivasan, "Distributed Q-learning neuro-type2 fuzzy system, " Submitted in IEEE Transactions on Neural Networks.
6. Balaji P.G and D.Srinivasan, "Modified symbiotic evolutionary learning for type-2 fuzzy system, " Submitted in International Journal on Fuzzy Systems.

MAGAZINE AND BOOK CHAPTERS

7. Balaji P.G and D. Srinivasan, "Multi-agent system in urban traffic signal control," in *IEEE Computational Intelligence Magazine*,vol.5, no.4,pp.43-51, 2010.
8. Balaji P.G and D. Srinivasan, "An introduction to multi-agent systems," in '*Innovations in Multi-Agent Systems and Applications', Studies on Computation Intelligence*, Springer, vol.310, pp.1-27, 2010.

CONFERNECES

9. Balaji P.G and D. Srinivasan, " Distributed multi-agent type-2 fuzzy architecture for urban traffic signal control," *IEEE International Conference on Fuzzy Systems*, pp. 1627-1632, 2009.
10. Balaji P.G, D. Srinivasan and C.K. Tham, "Coordination in distributed multi-agent system using type-2 fuzzy decision systems," *IEEE International Conference on Fuzzy Systems*, pp. 2291-2298, 2008.
11. Balaji P.G, G. Sachdeva, D. Srinivasan and C.K. Tham, "Multi-agent system based urban traffic management," *IEEE Congress on Evolutionary Computation*, pp.1740-1747, 2007.
12. Balaji P.G, D. Srinivasan and C.K. Tham, "Uncertainties reducing techniques in evolutionary computation," *IEEE congress on Evolutionary Computation*, pp.556-563, 2007

REFERENCES

[1] F. Webster, “Traffic Signal Settings,” *Road Research Technical Paper*, no. 39, 1958.

[2] B. Logan, and G. Theodoropoulos, “The distributed simulation of multiagent systems,” *Proceedings of the IEEE*, vol. 89, no. Copyright 2001, IEE, pp. 174-85, 2001.

[3] N. R. Jennings, K. Sycara, and M. Wooldridge, “A roadmap of agent research and development,” *Autonomous Agents and Multi-Agents Systems*, vol. 1, no. 1, pp. 7-38, 1999.

[4] L. C. Jain, and R. K. Jain, *Hybrid Intelligent Engineering Systems*, Singapore: World Scientific Publishing Company, 1997.

[5] C. Mumford, and L. C. Jain, *Computational Intelligence: Collaboration, Fusion and Emergence*: Springer-Verlag, 2009.

[6] L. C. Jain, M. Sato, M. Virvou *et al.*, *Computational Intelligence Paradigms: Volume 1 - Innovative Applications*: Springer-Verlag, 2008.

[7] L. C. Jain, and P. De Wilde, *Practical Applications of Computational Intelligence Techniques*, USA: Kluwer Academic Publishers, 2001.

[8] L. C. Jain, and N. M. Martin, *Fusion of Neural Networks, Fuzzy Logic and Evolutionary Computing and their applications*, USA: CRC Press, 1999.

[9] H. N. Tedorescu, A. Kandel, and L. C. Jain, *Fuzzy and Neuro-Fuzzy Systems in Medicine*, USA: CRC Press, 1998.

[10] J. Fulcher, and L. C. Jain, *Computational Intelligence: A Compendium*: Springer-Verlag, 2008.

[11] R. Khosla, N. Ichalkaranje, and L. C. Jain, *Design of Intelligent Multi-Agent Systems*: Springer-Verlag, 2005.

[12] G. Resconi, and L. C. Jain, *Intelligents Agents : Theory and Applications*: Springer-Verlag, 2004.

[13] L. C. Jain, Z. Chen, and N. Ichalkaranje, *Intelligent Agents and their Applications*: Springer-Verlag, 2002.

[14] L. Gasser, and M. Huhns, *Distributed Artificial Intelligence*: Morgan Kaufmann, 1989.

[15] K. P. Sycara, “The many faces of agents,” *AI magazine*, vol. 19, no. 2, pp. 11-12, 1998.

[16] T. Finin, C. Nicholas, and J. Mayfield. "Agent-based information retrieval tutorial," http://www.csee.umbc.edu/abir/.

[17] H. S. Nwana, “Software agents: an overview,” *Knowledge Engineering Review*, vol. 11, no. Copyright 1996, IEE, pp. 205-44, 1996.

[18] M. Woodridge, and N. R. Jennings, “Intelligent agents theory and practice,” *Knowledge Engineering Review,* vol. 10, no. Compendex, pp. 115-115, 1995.

[19] E. H. Durfee, and V. Lesser, "Negotiating Task ecomposition and Allocation Using Partial Global Planning," *Distributed Artificial Intelligence,* L. Gasser and M. Huhns, eds., pp. 229-244: Morgan Kaufmann, 1989.

[20] K. P. Sycara, “Multiagent Systems,” *AI Magazine,* vol. 19, no. 2, pp. 79-92, 1998.

[21] N. Vlassis, *Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence,* San Rafael, CA, USA: Morgan &amp; Calypool, 2007.

[22] P. Stone, and M. Veloso, “Multiagent systems: a survey from a machine learning perspective,” *Autonomous Robots,* vol. 8, no. Copyright 2000, IEE, pp. 345-83, 2000.

[23] Z. Ren, and C. J. Anumba, “Learning in multi-agent systems: a case study of construction claims negotiation,” *Advanced Engineering Informatics,* vol. 16, no. Copyright 2004, IEE, pp. 265-75, 2002.

[24] E. Alonso, M. D'Inverno, D. Kudenko *et al.*, “Learning in multi-agent systems,” *Knowledge Engineering Review,* vol. 16, no. Copyright 2002, IEE, pp. 277-84, 2001.

[25] C. V. Goldman, "Learning in multi-agent systems : A case study of construction claim negotiation." p. 1363.

[26] F. Bergenti, and A. Ricci, "Three approaches to the coordination of multiagent systems," *Proceedings of the ACM Symposium on Applied Computing.* pp. 367-373.

[27] C. H. Tien, and M. Soderstrand, "Development of a micro robot system for playing soccer games." pp. 149-152.

[28] P. G. Balaji, and D. Srinivasan, "Distributed multi-agent type-2 fuzzy architecture for urban traffic signal control," *2009 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE).* pp. 1627-32.

[29] L. E. Parker, “Heterogeneous multi-robot cooperation,” Massachusetts Institute of Technology, 1994.

[30] L. E. Parker, “Life-long adaptation in heterogeneous multi-robot teams: Response to Continual variation in robot performance,” *Autonomous Robots,* vol. 8, no. 3, 2000.

[31] R. Drezewski, and L. Siwik, "Co-evolutionary multi-agent system with predator-prey mechanism for multi-objective optimization," *Adaptive and Natural Computing Algorithms. 8th International Conference, ICANNGA*

2007. Proceedings, Part I (Lecture Notes in Computer Science Vol. 4431). pp. 67-76.

[32] A. Damba, and S. Watanabe, “Hierarchical control in a multiagent system,” *International Journal of Innovative Computing, Information &amp; Control,* vol. 4, no. Copyright 2009, The Institution of Engineering and Technology, pp. 3091-100, 2008.

[33] C. Min Chee, D. Srinivasan, and R. L. Cheu, “Neural networks for continuous online learning and control,” *IEEE Transactions on Neural Networks,* vol. 17, no. Copyright 2006, The Institution of Engineering and Technology, pp. 1511-31, 2006.

[34] P. G. Balaji, G. Sachdeva, D. Srinivasan *et al.*, "Multi-agent system based urban traffic management," *2007 IEEE Congress on Evolutionary Computation, CEC 2007.* pp. 1740-1747.

[35] A. Koestler, *The ghost in the machine*, London: Hutchinson Publication Group, 1967.

[36] P. Leitao, P. Valckenaers, and E. Adam, "Self-adaptation for robustness and cooperation in holonic multi-agent systems," *Transactions on Large-Scale Data- and Knowledge-Centered Systems. I,* pp. 267-88, Berlin, Germany: Springer-Verlag, 2009.

[37] O. Yadgar, S. Kraus, and C. Oritz, "Scaling up distributed sensor networks: Cooperative large scale mobile agent organizations," *Distributed Sensor Networks : A Multiagent Perspective,* pp. 267-288: LNCS 5740, 2003.

[38] M. Schillo, and F. Klaus, "A taxanomy of autonomy in multiagent organisation," *Autonomy 2003, LNAI 2969*, pp. 68-82, 2004.

[39] L. Bongearts, “Integration of scheduling and control in holonic manufacturing systems,” Katholieke Universiteit Leuven, 1998.

[40] D. Srinivasan, and M. Choy, "Distributed Problem Solving using Evolutionary Learning in Multi-Agent Systems," *Advances in Evolutionary Computing for System Design*, Studies in Computational Intelligence L. Jain, V. Palade and D. Srinivasan, eds., pp. 211-227: Springer Berlin / Heidelberg, 2007.

[41] M. Van De Vijsel, and J. Anderson, "Coalition formation in multi-agent systems under real-world conditions," *AAAI Workshop - Technical Report.* pp. 54-60.

[42] B. Horling, and V. Lesser, “A survey of multi-agent organizational paradigms,” *Knowledge Engineering Review,* vol. 19, no. Copyright 2006, IEE, pp. 281-316, 2004.

[43] A. K. Agogino, and K. Tumer, *Team Formation in Partially Observable Multi-Agent Systems*, United States, 2004.

[44] Budianto, *An overview and survey on multi-agent system*, 2005.

[45] C. Min Chee, D. Srinivasan, and R. L. Cheu, “Cooperative, hybrid agent architecture for real-time traffic signal control,” *IEEE Transactions on Systems, Man &amp; Cybernetics, Part A (Systems &amp; Humans),* vol. 33, no. Copyright 2003, IEE, pp. 597-607, 2003.

[46] P. G. Balaji, D. Srinivasan, and T. Chen-Khong, "Coordination in distributed multi-agent system using type-2 fuzzy decision systems," *2008 IEEE 16th International Conference on Fuzzy Systems (FUZZ-IEEE).* pp. 2291-8.

[47] S. E. Lander, “Issues in multiagent design systems,” *IEEE Expert,* vol. 12, no. Copyright 1997, IEE, pp. 18-26, 1997.

[48] J.-S. Lin, C. Ou-Yang, and Y.-C. Juan, “Towards a standardised framework for a multi-agent system approach for cooperation in an original design manufacturing company,” *International Journal of Computer Integrated Manufacturing,* vol. 22, no. Compendex, pp. 494-514, 2009.

[49] Y. Cengeloglu, “A framework for dynamic knowledge exchange among intelligent agents,” in AAAI Symposium, Control of the Physical World by Intelligent Agents, 1994.

[50] M. Genesereth, and R. Fikes, *Knowledge Interchange Format, Version 3.0 Reference Manual,* Computer Science Department, Stanford University, USA, 1992.

[51] M. L. Ginsberg, “Knowledge interchange format: the KIF of death,” *AI Magazine,* vol. 12, no. Copyright 1992, IEE, pp. 57-63, 1991.

[52] T. Finin, R. Fritzson, D. McKay *et al.*, "KQML as an agent communication language," *CIKM 94. Proceedings of the Third International Conference on Information and Knowledge Management.* pp. 456-63.

[53] A. Greenwald, *The search for equilibrium in markov games*: Synthesis Lectures on Artificial Intelligence and Machine Learning, 2007.

[54] Y. M. Ermol'ev, and S. P. Uryas'ev, “Nash equilibrium in n-person games,” *Cybernetics,* vol. 18, no. Copyright 1983, IEE, pp. 367-72, 1982.

[55] R. Gibbons, “An introduction to applicable game theory,” *The Journal of Economic Perspectives,* vol. 11, no. 1, pp. 127-149, 1997.

[56] H. Nwana, L. Lee, and N. Jennings, "Co-ordination in multi-agent systems," *Software Agents and Soft Computing Towards Enhancing Machine Intelligence,* Lecture Notes in Computer Science H. Nwana and N. Azarmi, eds., pp. 42-58: Springer Berlin / Heidelberg, 1997.

[57] A. Chavez, and P. Maes, "Kasbah: an agent marketplace for buying and selling goods," *Acquisition, Learning and Demonstration: Automating Tasks for Users. Papers from the 1996 AAAI Symposium (TR SS-96-02).* pp. 8-12.

[58] L. Kuyer, S. Whiteson, B. Bakker *et al.*, "Multiagent reinforcement learning for Urban traffic control using coordination graphs," *Lecture Notes in*

Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 656-671.

[59] R. G. Smith, “The contract net protocol: high level communication and control in a distributed problem solver,” *IEEE Transactions on Computers,* vol. C-29, no. Copyright 1981, IEE, pp. 1104-13, 1980.

[60] P. D. O'Brien, and R. C. Nicol, “FIPA-towards a standard for software agents,” *BT Technology Journal,* vol. 16, no. Copyright 1998, IEE, pp. 51-9, 1998.

[61] C. Guestrin, S. Venkataraman, and D. Koller, "Context-specific multiagent coordination and planning with factored MDPs," *Proceedings of the National Conference on Artificial Intelligence.* pp. 253-259.

[62] D. Srinivasan, C. Min Chee, and R. L. Cheu, “Neural networks for real-time traffic signal control,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 7, no. Copyright 2006, The Institution of Engineering and Technology, pp. 261-72, 2006.

[63] L. Lhotska, "Learning in multi-agent systems: theoretical issues," *Computer Aided Systems Theory - EUROCAST '97. Selection of Papers from the 6th International Workshop on Computer Aided Systems Theory. Proceedings.* pp. 394-405.

[64] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Efficient non-linear control through neuro evolution." pp. 654-662.

[65] J. Jiu, *Autonomous Agents and Multi-Agent Systems*: World Scientific Publication.

[66] V. Vassiliades, A. Cleanthous, and C. Christodoulou, "Multiagent Reinforcement Learning with Spiking and Non-Spiking Agents in the Iterated Prisoner’s Dilemma," *Artificial Neural Networks – ICANN 2009,* Lecture Notes in Computer Science C. Alippi, M. Polycarpou, C. Panayiotou *et al.*, eds., pp. 737-746: Springer Berlin / Heidelberg, 2009.

[67] T. Gabel, and M. Riedmiller, "On a successful application of multi-agent reinforcement learning to operations research benchmarks," *2007 First IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (IEEE Cat. No.07EX1572).* p. 8 pp.

[68] R. S. Sutton, and A. G. Barto, *Reinforcement Learning: An Introduction,* Cambridge, MA: MIT Press.

[69] J. Schneider, W. Weng-Keen, A. Moore *et al.*, "Distributed value functions," *Machine Learning. Proceedings of the Sixteenth International Conference (ICML'99).* pp. 371-8.

[70] L. Busoniu, R. Babuka, and B. De Schutter, “A comprehensive survey of multiagent reinforcement learning,” *IEEE Transactions on Systems, Man and*

Cybernetics Part C: Applications and Reviews, vol. 38, no. Compendex, pp. 156-172, 2008.

[71] C. J. Messer, H. E. Haenel, and E. A. Koeppe, *A Report on the User's Manual for Progression Analysis and Signal System Evaluation Routine--Passer II*, United States, 1974.

[72] E. C. P. Chang, S. L. Cohen, C. Liu *et al.*, “MAXBAND-86. Program for optimizing left-turn phase sequence in multiarterial closed networks,” *Transportation Research Record*, no. Compendex, pp. 61-67, 1988.

[73] C. Stamatiadis, and N. H. Gartner, “MULTIBAND-96: A program for variable-bandwidth progression optimization of multiarterial traffic networks,” *Transportation Research Record*, no. Compendex, pp. 9-17, 1996.

[74] C. J. Messer, and M. P. Malakapalli, *Applications Manual for Evaluating Two and Three-Level Diamond Interchange Operations Using Transyt-7F*, United States, 1992.

[75] C. Sun, and J. Xu, “Study on Traffic Signal Timing Optimization for Single Point Intersection Based on Synchro Software System,” *Journal of Highway and Transportation Research and Development,* vol. 26, no. Copyright 2010, The Institution of Engineering and Technology, pp. 117-22, 2009.

[76] S. R. Sunkari, R. J. Engelbrecht, and K. N. Balke, *Evaluation of advanced coordination features in traffic signal controllers*, FHWA, September, 2004.

[77] M. C. Bell, and R. D. Bretherton, "Ageing of fixed-time traffic signal plans."

[78] K. Fehon, “Adaptive traffic signals are we missing the boat,” in ITE District 6 Annual Meeting, 2004.

[79] P. R. Lowrie, "The Sydney Coordinated Adaptive Traffic System-principles, methodology, algorithms," *International Conference on Road Traffic Signalling.* pp. 67-70.

[80] D. I. Robertson, and R. D. Bretherton, “Optimizing networks of traffic signals in real time-the SCOOT method,” *IEEE Transactions on Vehicular Technology,* vol. 40, no. Copyright 1991, IEE, pp. 11-15, 1991.

[81] P. B. Hunt, D. I. Robertson, R. D. Bretherton *et al.*, *SCOOT-A traffic responsive method for coordinating signals*, TRL, 1981.

[82] F. Busch, and G. Kruse, "MOTION for SITRAFFIC - A modern approach to urban traffic control," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC.* pp. 61-64.

[83] C. Bielefeldt, and F. Busch, "MOTION-a new on-line traffic signal network control system," *Seventh International Conference on `Road Traffic Monitoring and Control' (Conf. Publ. No.391).* pp. 55-9.

[84] M. Papageorgiou, "An introduction to signal traffic control strategy TUC."

[85] V. Mauro, and C. Di Taranto, "UTOPIA [traffic control]," *Control, Computers, Communications in Transportation. Selected Papers from the IFAC/IFIP/IFORS Symposium.* pp. 245-52.

[86] N. H. Gartner, S. F. Assmann, F. Lasaga *et al.*, "A multi-band approach to arterial traffic signal optimization," *Transportation Research, Part B (Methodological),* vol. 25B, no. Copyright 1991, IEE, pp. 55-74, 1991.

[87] N. H. Gartner, J. D. C. Little, and H. Gabbay, "Simultaneous optimization of offsets, splits, and cycle time," *Transportation Research Record,* no. Compendex, pp. 6-15, 1976.

[88] N. H. Gartner, F. J. Pooran, and C. M. Andrews, "Implementation of the OPAC adaptive control strategy in a traffic signal network," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC.* pp. 195-200.

[89] J. F. Barriere, J. L. Farges, and J. J. Henry, "Decentralization vs hierarchy in optimal traffic control," *IFAC Proceedings Series.* pp. 209-214.

[90] J. L. Farges, I. Khoudour, and J. B. Lesort, "PRODYN: on site evaluation," *Third International Conference on Road Traffic Control (Conf. Publ. No.320).* pp. 62-6.

[91] P. Mirchandani, and L. Head, "A real-time traffic signal control system: Architecture, algorithms, and analysis," *Transportation Research Part C: Emerging Technologies,* vol. 9, no. Compendex, pp. 415-432, 2001.

[92] K. L. Head, P. B. Mirchandani, and D. Sheppard, "Hierarchical framework for real-time traffic control," *Transportation Research Record,* vol. 1360, pp. 82-88, 1992.

[93] P. Dell'Olmo, and P. B. Mirchandani, "REALBAND: an approach for real-time coordination of traffic flows on networks," *Transportation Research Record,* no. Compendex, pp. 106-116, 1995.

[94] Sen.S., and K. L. Head, "Controlled optimization of phases at an intersection," *Transportation Science,* vol. 3, pp. 5-17, 1997.

[95] L. A. Zadeh, "Concept of a linguistic variable and its application to approximate reasoning - 1," *Information Sciences,* vol. 8, no. Compendex, pp. 199-249, 1975.

[96] L. A. Zadeh, "Concept of a linguistic variable and its application to approximate reasoning - 2," *Information Sciences,* vol. 8, no. Compendex, pp. 301-357, 1975.

[97] L. A. Zadeh, "Concept of a linguistic variable and its application to approximate reasoning - 3," *Information Sciences,* vol. 9, no. Compendex, pp. 43-80, 1975.

[98] L. Qilian, and J. M. Mendel, "Interval type-2 fuzzy logic systems: theory and design," *IEEE Transactions on Fuzzy Systems,* vol. 8, no. Copyright 2000, IEE, pp. 535-50, 2000.

[99] N. N. Karnik, J. M. Mendel, and L. Qilian, "Type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems,* vol. 7, no. Copyright 2000, IEE, pp. 643-58, 1999.

[100] N. N. Karnik, and J. M. Mendel, "Centroid of a type-2 fuzzy set," *Information Sciences,* vol. 132, no. Copyright 2001, IEE, pp. 195-220, 2001.

[101] W. Hongwei, and J. M. Mendel, "Introduction to uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems," *10th IEEE International Conference on Fuzzy Systems. (Cat. No.01CH37297).* pp. 662-5.

[102] S. Coupland, and R. John, "New geometric inference techniques for type-2 fuzzy sets," *International Journal of Approximate Reasoning,* vol. 49, no. Compendex, pp. 198-211, 2008.

[103] Z. Wang, and J.-x. Fang, "On the direct decomposability of pseudo-t-norms, t-norms and implication operators on product lattices," *Fuzzy Sets and Systems,* vol. 158, no. Compendex, pp. 2494-2503, 2007.

[104] S. Coupland, and R. John, "Geometric type-1 and type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems,* vol. 15, no. Copyright 2007, The Institution of Engineering and Technology, pp. 3-15, 2007.

[105] J. Pach, and M. Sharir, "On vertical visibility in arrangements of segments and the queue size in the Bentley-Ottmann line sweeping algorithm," *SIAM Journal on Computing,* vol. 20, no. Compendex, pp. 460-470, 1991.

[106] W. Hongwei, and J. M. Mendel, "Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems,* vol. 10, no. Copyright 2002, IEE, pp. 622-39, 2002.

[107] J. Niittymaki, "General fuzzy rule base for isolated traffic signal control-rule formulation," *Transportation Planning and Technology,* vol. 24, no. Compendex, pp. 227-247, 2001.

[108] C. Min Chee, "Cooperative, hybrid multi-agent system for distributed , real-time traffic signal control," Department of Electrical and Computer Engineering, National University of Singapore, Singapore, 2006.

[109] D. E. Moriarty, and R. Miikkulainen, "Efficient learning from delayed rewards through symbiotic evolution," *Machine Learning. Proceedings of the Twelfth International Conference on Machine Learning.* pp. 396-404.

[110] D. E. Moriarty, and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Machine Learning,* vol. 22, no. Copyright 1996, IEE, pp. 11-32, 1996.

[111] B. H. G. Barbosa, L. T. Bui, H. A. Abbass *et al.*, “The use of coevolution and the artificial immune system for ensemble learning,” no. Compendex, pp. 1-13, 2010.

[112] G. P. Figueredo, L. A. V. de Carvalho, and H. J. C. Barbosa, "Coevolutionary genetic algorithms to simulate the immune system's gene libraries evolution," *Advances in Natural Computation. First International Conference, ICNC 2005. Proceedings, Part II (Lecture Notes in Computer Science Vol. 3611).* pp. 941-4.

[113] Z. Qiuyong, R. Jing, Z. Zehua *et al.*, "Immune co-evolution algorithm based on chaotic optimization," *2007 Workshop on Intelligent Information Technology Application.* pp. 149-52.

[114] C.-F. Juang, and C.-T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE International Conference on Fuzzy Systems.* pp. 1281-1285.

[115] Y.-C. Hsu, S.-F. Lin, and Y.-C. Cheng, “Multi groups cooperation based symbiotic evolution for TSK-type neuro-fuzzy systems design,” *Expert Systems with Applications,* vol. 37, no. Compendex, pp. 5320-5330, 2010.

[116] M. Mahfouf, M. Jamei, and D. A. Linkens, "Rule-base generation via symbiotic evolution for a mamdani-type fuzzy control system," *IEEE International Conference on Fuzzy Systems.* pp. 396-399.

[117] F.-z. Yi, H.-z. Hu, and D. Zhou, “Fuzzy controller auto-design based on the symbiotic evolution algorithm,” *Systems Engineering and Electronics,* vol. 25, no. Copyright 2004, IEE, pp. 750-3, 2003.

[118] H. B. Kazemian, “Study of learning fuzzy controllers,” *Expert Systems,* vol. 18, no. Copyright 2001, IEE, pp. 186-93, 2001.

[119] K. Tanaka, T. Taniguchi, and H. O. Wang, "Generalized Takagi-Sugeno fuzzy systems: rule reduction and robust control," *Ninth IEEE International Conference on Fuzzy Systems. FUZZ- IEEE 2000 (Cat. No.00CH37063).* pp. 688-93.

[120] P. Liu, “Mamdani fuzzy system: Universal approximator to a class of random processes,” *IEEE Transactions on Fuzzy Systems,* vol. 10, no. Compendex, pp. 756-766, 2002.

[121] C. Lynch, H. Hagras, and V. Callaghan, "Using uncertainty bounds in the design of an embedded real-time type-2 neuro-fuzzy speed controller for marine diesel engines," *IEEE International Conference on Fuzzy Systems.* pp. 1446-1453.

[122] C. J. C. H. Watkins, “Learning from delayed rewards,” University of Cambridge, 1989.

[123] R. A. Jacobs, “Increased Rates of Convergence Through Learning Rate Adaptation,” *Neural Networks,* vol. 1, no. Compendex, pp. 295-307, 1988.

[124] R. T. Van Katwijk, P. Van Koningsbruggen, B. De Schutter *et al.*, "Test bed for multiagent control systems in road traffic management," *Transportation Research Record.* pp. 108-115.

[125] S. Mikami, and Y. Kakazu, "Genetic reinforcement learning for cooperative traffic signal control," *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence (Cat. No.94TH0650-2).* pp. 223-8.

[126] L. Jee-Hyong, and L.-K. Hyung, "Distributed and cooperative fuzzy controllers for traffic intersections group," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews),* vol. 29, no. Copyright 1999, IEE, pp. 263-71, 1999.

[127] M. B. Trabia, M. S. Kaseko, and M. Ande, "A two-stage fuzzy logic controller for traffic signals," *Transportation Research Part C (Emerging Technologies),* vol. 7C, no. Copyright 2000, IEE, pp. 353-67, 1999.

[128] S. Chiu, and S. Chand, "Self-organizing traffic control via fuzzy logic," *Proceedings of the 32nd IEEE Conference on Decision and Control (Cat. No.93CH3307-6).* pp. 1897-902.

[129] Quadstone, *PARAMICS Modeller v6.0 User Guide and Reference Manual*, Quadstone Ltd, Edinburgh, UK, 2002.

[130] J. Little, "A Proof for the Queuing Formula: L= λ W," *Operations Research,* vol. 9, no. 3, pp. 383-387, 1961.

[131] J. R. Peirce, and P. J. Webb, "MOVA control of isolated traffic signals-recent experience," *Third International Conference on Road Traffic Control (Conf. Publ. No.320).* pp. 110-13.

[132] T. R. Board, "Highway Capacity Manual," National Research Council, 2000.