Decentralized Artificial Intelligent Traffic Control

Heung Tsan Hing

M. PHIL.

THE HONG KONG POLYTECHNIC UNIVERSITY

2001

Abstract

Abstract of thesis entitled 'Decentralized Artificial Intelligent Traffic Control' submitted by Heung Tsan Hing for the degree of M. PHIL. at the Hong Kong Polytechnic University in December 2000.

A generic decentralized traffic junction controller using artificial intelligence is constructed. The controller is simple in structure and does not require any geographic knowledge of the junction. Each junction has a decentralized traffic junction controller that can be connected together to form traffic network control. Number of controllers can join together to form a large network.

The controller consists of 3 modules, a local decision module, an optimization module and a junction coordination module. The local decision module is a fuzzy controller with a hierarchical structure. The optimization module use Genetic Algorithms to generate fuzzy rules for the local decision module. The junction coordination module adopts real time dynamic programming to obtain coordinated result. Several optimization techniques are used to enhance the performance of the controller, such as hierarchy and time alignment.

Cross-junction with turning traffic is modeled. Computer simulation is used to evaluate the performance of the controller against fixed-time approach. The controller can improve up to 20% and 8% for single and multiple junction environments respectively. Other approaches that can further improve the controller are also discussed at the end.

Acknowledgements

The author would like to take this opportunity to thank Dr. T. K. Ho for his invaluable guidance, kind continuous encouragement throughout the project.

The author would also like to express his sincere thanks for his colleagues for their friendships and encouragement.

The financial assistance given by the Research Committee and Electrical Engineering Department of The Hong Kong Polytechnic University through the award of research studentship and funding for international conference attendance is also greatly appreciated.

SYNOPSIS

Road network plays a key role in our daily life. Everyday tremendous amount of time is spent on traveling, to school and to work, and distribution of goods to local retail stores. Surveys carried out by Transport Department [4] shows that the traffic demand in Hong Kong is actually growing yearly. It is a common knowledge that, road capacity is limited. We cannot allow the traffic demand growing without any control. Otherwise, the network will be saturated and hence area-wide traffic congestion will occur. There are so many ways in suppressing the traffic growth or reducing traffic congestion. For example, increasing the tax levy for car ownership can reduce the growth of vehicle. Introduction of road pricing scheme can discourage drivers from going in to the area. Building more roads to enhance network capacity. Redesign of the network flow may help redirecting vehicles to other high capacity road. All the above methods can reduce or even suppress traffic growth. However, no one single way can guarantees success. It is because so many factors such as the budget constraint, physical limitation and also social impact have to be included in making of the policy,. As a result, traffic control is as much art as science.

There is no scientific way to deal with traffic control? Not really. On macroscopic policy definition, there is no scientific way. However, for microscopic view, there is. A road network consists of many roads interconnecting each other. At each intersection, there must be some control method applied to, no matter it is a simple policy or a controller physically attached. If every single junction has to be considered in a macroscopic policy definition, then there will be too much. As a result, individual junction control mechanism will be left out until the overall policy is defined. The individual junction control can then be defined using standard methods based on the estimated traffic flow. In this study, only the microscopic view is covered. We intended to create a simple and generic controller in order

to simplify the design and deployment of a junction controller.

The prime objective of area traffic control is to minimize number of stops of vehicles as well as average delay imposed on vehicles within the network. This can be done by calculating the red-green signal cycle time of individual junctions and the offsets between adjacent junctions. Each junction has its own characteristics and time-varying traffic conditions; formulation of optimal cycle times for different junctions may be different. On the other hand, the ideal signal offset between adjacent junctions is that no vehicles need to stop when passing through the junction. The signal offset is then equal to the traveling time between adjacent junctions. This is relatively simple for a one-way linear network. However the problem becomes complicated for a realistic two-way network and when the cycle time of two adjacent junctions is not equal. When the offset is optimized for one direction, the other direction may be heavily penalized or vice versa. Besides, the different cycle times in the junctions will divert the co-ordination between them. The road network has therefore to be considered as a whole to achieve a global optimum.

In this study, a decentralized traffic controller using artificial intelligence techniques has been designed and built. It consists of 3 modules, a local decision module, an optimization module, and a junction coordination module. The local decision module determines the effective green time for each individual road junction using rules generated by the optimization module. The junction coordination module adjusts the effective green time to attain the network optimum. The controllers are identical for each individual junction that allows two-way traffic with turnings. Fuzzy logic, genetic algorithms and dynamic programming are used in constructing the controller. A number of tests have been carried out to assess the performance of the controller under different traffic conditions. It has been shown that for a single traffic junction, the controller reduces the traffic delay by 24% when compared with a commonly used fixed-time traffic controller. Under a simple network, the controller can still achieve 8% reduction on average.

Table of Contents

1 Introduction

Modern traffic control at road junction is divided into two main categories, fixed-time and flow-responsive control. Fixed-time control is based on traffic data obtained from road survey, it then calculates the proper signal cycle time and offset plan for the network. Since the control action is per-determined, the performance is fairly stable under steady traffic condition. However, when there are unexpected and abrupt changes in traffic condition, this approach fails to solve the problem satisfactorily, and sometimes even leads to aggravation. Nowadays, the availability of vehicle detectors makes it possible to obtain real-time traffic data and hence control action can be made accordingly.

Flow-responsive control obtains real time traffic data from sensors installed at appropriate location and uses them to calculate the cycle time and offset plan. As a result, it is capable of dealing with time changing traffic condition. Moreover, traffic data obtained from the sensors can replace the time-consuming road traffic survey; which is important to traffic engineers in designing and maintaining the road network.

In area traffic control, the prime objective is to calculate signal cycle time and offset between adjacent junctions so as to minimize the number of stops the vehicles made, as well as the delay of vehicles suffered in the network. Each junction has different characteristics and traffic conditions, so its requirements on cycle time are different from others. The ideal signal offset between adjacent junctions is that no vehicles need to stop when moving from one junction to another. The signal offset is then equal to the traveling time between two junctions. This is simple for a single-directional flow network with no turning traffic.

However, the problem becomes complicated if turning and bi-directional traffic flow is allowed. If the offset is optimized for one direction, the other direction may suffer severe delay or vice versa. As a result, the network has to be considered as a whole to obtain a global optimum. Alternatively, large network is divided into blocks and a near optimum is obtained for each block.

In this study, a traffic controller has been designed and built by artificial intelligence techniques. The traffic controller consists of 3 modules, a local decision module, an optimization module and a junction coordination module. The local decision module and the optimization module form a hierarchical fuzzy logic controller. The purpose is to determine signal cycle plan for each individual road junction. The junction coordination module enables communication with adjacent junctions and adjusts the signal time to produce a network optimum.

The local decision module is designed to calculate individual junction's signal time. The unit makes use of local information, such as present queue length and flow-rates, to determine effective green time of each signal phase of the junction. The hierarchical fuzzy logic controller divides the decision-making process into 2 layers. The first layer takes queue length and flow-rate as input; it estimates the effective green time required for a particular phase. The second layer takes the flow-rate of one link and overall junction flow-rates as input. It considers the whole junction saturation and adds an adjustment term to the result of the first layer. Human expert determines the rule-base for the first layer of control. The optimization module, make use of genetic algorithms, determines the second layer rules.

The junction coordination module is designed to deal with the coordination between adjacent junctions. It employs dynamic programming to determine a set of control actions that minimize the system's penalty function, which is a weighed sum of delays of the vehicles and number of stops of every signal junction in the network. Each phase of the signal cycle represents a stage of the system. Provided a set of coordination control (stage transitions), the system evolves from one stage to another, that is from one signal phase to another. After building a state-space transformation diagram, an optimal path of the system can be determined by using dynamic programming. The path between states represents the coordination adjustment parameter of each phase. The final green time duration for a signal phase is the sum of effective green time obtained by the local decision module and the coordination adjustment parameter.

This thesis is divided into 3 sections, literature review, modeling, and results. The literature review contains theories, traditional implementation of traffic controller, and introduction to technologies used. The modeling section describes the gradual progression in constructing the controller. The results section expresses the simulation results and discussion on the results achieved.

2 Road Junction Control

2.1 Overview

Road network plays a key role in our daily life. Everyday tremendous amount of time is spent on traveling, to school and to work, and distribution of goods to local retail stores. From an economic point of view, traffic congestion degrades quality of life in general, especially in a busy city like Hong Kong where time is very important. From environmental point of view, unnecessary vehicle stops will produce more exhaust gases, which damages our health. From engineering point of view, the efficiency of natural energy needs to be maximized by eliminating unnecessary stop and shortening the travel time. Effective traffic control can minimize the total time delay by properly allocating the right-of-way among all road users.

Traffic controller is installed at road junctions where two or more traffic streams intersect. The controller assigns the right-of-way to one or more traffic stream in turn. The junction itself is a common area where no two contradictory traffic streams are using it at the same time. For example, a vehicle traveling from north to south direction cannot enter the junction when the east-west direction is given the right-of-way. Otherwise, traffic accident will occur. Traffic control is to utilize the use of the common area such that it is fairly distributed among all the traffic streams. Optimum control means average delay is minimized among all the junction users. Proper setting of time and duration of the red-amber-green signals can do this. The same argument can be applied to traffic control over an area network.

Khalili and Macleod [1,2] argued that when traffic at the junctions of an urban road network is optimized locally, a global optimization could be achieved by coordinating the junctions with respect to each other. They achieved such coordination by optimizing the time offset of common periods between two adjacent junctions. The optimization aimed to minimize delays, stops or weighted combination of the two of the vehicles. In this chapter, some currently adopted methods will be given here to provide a brief understanding of single junction control.

Single junction control can be divided into 2 main categories, fixed-time and traffic responsive control. Fixed-time control is an offline optimization process while the traffic responsive control is real time optimization.

2.2 Fixed-Time Control

With fixed-time signals, the sequence of lights shown on each approach to the intersection has a cycle of fixed duration, and each signal indication appears for a fixed period of time. A fixed-time control policy is a static optimization, which is related to the historical flow parameters.

A fixed-time control system is simple in structure and does not require dynamic vehicle detection. It relies on the historical data to prepare timing plans for a signalized area. Since flow pattern of one-day traffic is quite similar, it can be divided into three to four sections representing the morning peak, afternoon peak, off-peak, etc. A timing plan for each section is formulated and put into the controller, which switches the plans according to the time of a day. These timing plans are determined off-line and cannot be adjusted frequently.

The lack of real time traffic data is a major drawback of a fixed-time controller. The controller fails to respond in changes of traffic volume. Hence, the fixed-time controller needs to be re-calibrated frequently after a period of time. Moreover, traffic engineers have to monitor the controller regularly to ensure the controller is capable of handling the traffic without imposing unnecessary delay to the traffic.

Determination of control plans is based on collection and projection of traffic data. Either human or automatic roadside detectors will collect the data first. Then it will be divided into different periods based on traffic volume, usually peak and off-peak. Since it is not practical to calculate the cycle-time every day, a constant growth of traffic volume with linear increment is assumed. This is to ensure the cycle plan is adequate to deal with traffic growth over a period of time.

TRANSYT [6,34] is the most commonly used package to determine fixed-time control plan. It is an offline optimization system that calculates cycle-time, phase split and offset plan for a road network by using historical traffic data from human count or automatic roadside detectors.

2.3 Traffic Responsive Control

Traffic-responsive control is an online control that takes real time traffic data as input and optimizes the output control action according to the prevailing traffic conditions. Unlike the fixed-time control, an online policy eliminates the discrepancy of traffic conditions with extra hardware cost, mainly for detectors. It has the ability to adjust itself to suit the traffic variations in real time due to its predictive nature and also the selection of the desired function of minimization of stops, delays, fuel consumption, or exhaust emission rate for the whole network or sub-areas within the network [3]. Moreover, a well-defined traffic-responsive control can become a fully automatic system that minimizes the time of human interaction.

Early research in traffic adaptive control, such as RTOP in Toronto [32], ASCOT in San Jose [35] and CYRANO in Washington, DC [41] concentrated on flow or demand prediction. The philosophy was to predict the demand in one or several cycle times in the future and to optimize signal settings according to the predicted demand. However, the difficulty in flow prediction leads to the failure of the early approaches.

Another traffic responsive method calculates control parameters according to the prevailing traffic conditions. It dynamically adjusts cycle time and phase split. Luk [28] compared the two traffic responsive control methods, Sydney Coordinated Adaptive Traffic method (SCAT) and Split, Cycle and Offset Optimization Technique (SCOOT). Both methods are now in daily operation and both have shown better performance than the fixed-time control. However, there are periods during which fixed-time control actually out-performed either SCAT or SCOOT. Luk emphasized that response time of a controller should be short, preferably no more than one cycle time. It is due to the highly fluctuating traffic flow from one cycle time to another. Parameters measured in a cycle, weighted together with data obtained in previous cycles, should be used to implement control strategies for the following cycle.

Apart from SCAT and SCOOT, attempts have been made to apply different control methodologies in traffic control, especially in advanced artificial intelligent (AI) technique. When a traffic warden coordinates traffic when the traffic signal malfunctions, he may not know how to model the junction and calculate the cycle time, but he can still make good judgment using his experience. The AI technique is based on the similar concept. It presents human experience and knowledge by a set of rules. Fuzzy logic is one of widely used AI technique to model those rules. Elahi, Radwan and Goul [9] switches control between fixed-time and actuated strategy according to flow-rate of a junction. Pappis and Mamdani [30] are pioneers of applying fuzzy logic in junction control. Chiu and Chand [6] extended Pappis's study by introducing more complex rules into the system. Ho [17] further extended fuzzy logic control and showed that further reduction of the average delay is possible when the fuzzy rules are adapted to the current traffic conditions. Spall and Chin [37], F.S. Ho and Ioannou [16] apply artificial neural network in junction control. They use neural network to predict flow-rates and hence determine proper control action for the junction. Findler and his colleague [10] implemented a self-learning knowledge based control system.

With the exception of Findler and his colleague's work, all the above studies only investigate simple junction with no turning traffic. The approach is applicable when traffic is light and the vehicle turning percentage is small. However, when traffic is heavy, an extra phase must be provided to clear the turning vehicles.

2.4 Single Junction Control

A brief introduction on terms and cycle time calculation of an isolated road junction is given here.

A control cycle consists of a number of phases. Each phase allows one particular traffic stream passing through the junction. For a simple cross-junction without turning traffic, a complete control cycle has two phases allowing two incoming traffic streams to pass through.

2.4.1 Effective Green Time and Saturation Flow

Figure 2.1 and 2.2 are obtained from Webster [42] which show the relationship between distance time and discharge rate of a signal-controlled junction. When green period commences, the vehicles are accelerating to normal running speed in first few seconds, as shown in Figure 2.1. The queue discharges at a more or less constant rate afterward, called the saturation flow. If there is still a queue at the end of the green period some vehicles will make use of the amber period to cross the intersection. Under these circumstances, traffic moves on both green and amber signals but the discharge rate is less than the saturation flow both at the beginning and at the end of the right-of-way period, as shown in Figure 2.2.

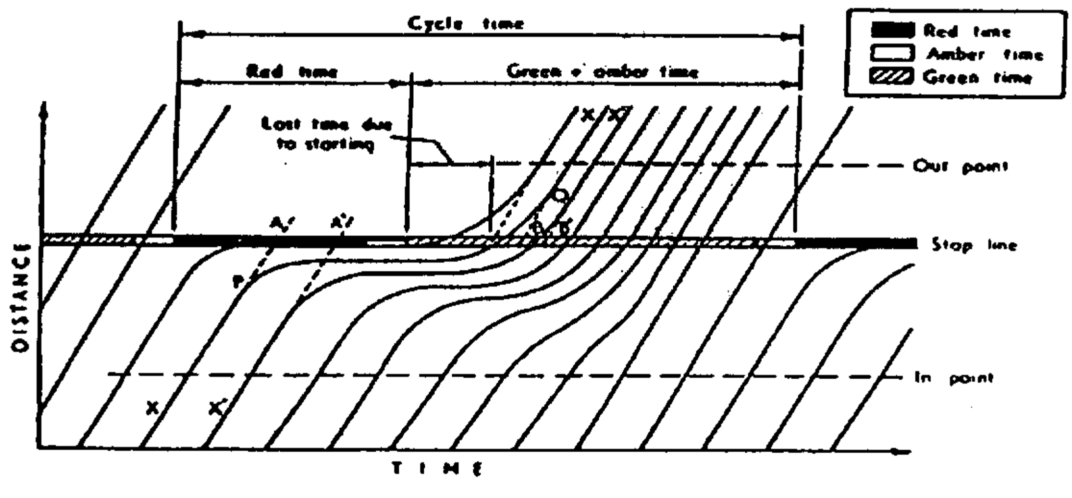

Figure 2.1 Distance/time diagram for signal-controlled junction obtained in Webster [42].

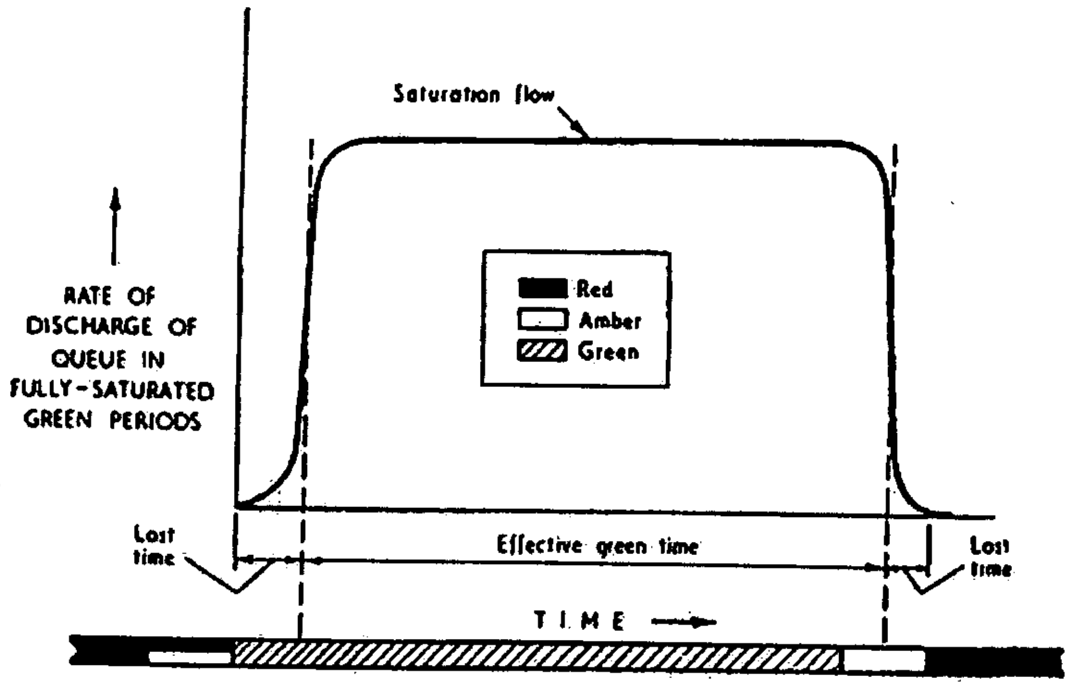

Figure 2.2 Variation of discharge rate of queue with time in fully saturated green period obtained in Webster [42].

The green and amber periods ($k + a$) may be replaced by an "effective" green (g) and a "lost" time (l). The product of the effective green and the saturation flow is equal to the maximum number of vehicles (say, b) discharged from the queue in a saturated green period (i.e. a green period during which the queue never clears).

$$k + a = g + l$$
$$b = gs$$

Where s is the saturation flow.

2.4.2 Optimal Cycle Time

Most control methodologies calculate cycle time of a junction. The optimum cycle time can be approximated by Webster's equation [42].

Equation 2.1 Webster optimal cycle time equation.

$$c_o = \frac{1.5L + 5}{1 - Y} \text{ seconds}$$

Where Y is the sum of flow rate ratio of the junction, and

L is the total lost time per cycle.

Equation 2.2 Total lost time.

$$L = nl + R$$

Where n is the number of phases,

l is the average lost time per phase, and

R is the time during each cycle when all signals display red.

2.4.3 Green Time of a Phase

The optimum cycle time calculation assumes that the effective green time of the phase is in the ratio of its flow to the junction saturation flow. After calculating the optimum cycle time, the green time of phase i, k_i, can be obtained by equation 2.3.

Equation 2.3 Effective green time of a phase.

$$k_i = (c_o - L)\frac{f_i}{\sum_{j=1}^{n} f_j} \text{ seconds}$$

Where f_i is the flow to saturation flow ratio in phase i, n equals to the number of traffic streams of the junction.

2.5 Area Traffic Control

A traffic network consists of many interconnected road junctions. Area traffic control is to apply some coordination control between junctions such that overall average delay and the number of stops per vehicle in the area are minimized.

At the boundaries of a traffic network, vehicles arrive in random. Within the network, vehicles tend to form platoons at junctions and move from one junction to another. Aligning common green time of adjacent junctions can minimize the delay. That is to find an optimum offset within adjacent junctions.

There are two approaches in controlling groups of junctions, centralized and decentralized control. Centralized control gathers all information from different area and computes signal plans for each junction. The computation in centralized approach can be carried out either on-line or off-line. On the other hand, local controller is used in decentralized control. All information, including some coordination parameters from adjacent junction, is processed locally without considering the whole network at the time. Decentralized control is suitable for on-line, real time control because less computational effort is needed.

2.5.1 Centralized Control

The centralized approaches use a higher-level controller coordinating the junction controllers. SCAT is a typical example. It divides the whole area into a number of sub-areas. In each sub-area, a controller coordinates junctions. Between sub-areas, hard coded time split is used for synchronization. Luk [28] compared SCAT, SCOOT and fixed-time control and found that SCAT performs well inside sub-area, field survey found that it is 2% better than fixed-time control in journey time and 1% better in stops. However, during peak hour at lunchtime, it is 6% worse than the fixed-time. One major limitation of SCAT is that the coordination between sub-areas is predetermined. The lack of feedback in sub-area coordination may introduce unwanted traffic congestion along the boundary of sub-area especially during peak hour. The boundary traffic congestion will spread out and cause area-wise traffic congestion.

TRANSYT is another typical example of offline-centralized control. It optimizes the whole network globally by considering junction cycle time, phase split and junction offset all together. However, since it is an offline control, any changes in traffic condition will degrade the result calculated.

Since centralized control takes the whole network or sub-network into account at a time, it is possible to achieve global optimum. However, the time for computing the network optimal is directly proportional to the size of the network. It is particularly significant in flow-responsive control, where the response time of the controller is critical.

2.5.2 Decentralized Control

The decentralized approach is different from the centralized approach. Instead of dividing an area into a number of sub-areas, the decentralized approach considers each single junction as an individual unit. Every junction controller is responsible for the coordination with its adjacent junctions by exchanging information like current phase, green time length and queue lengths. After gathering all information from adjacent junctions, the controller will make decision on the extension of green period. Heydecker [15] provided a systematic method for constructing decentralized control, starting from designing the signal sequence. Findler and his colleagues [10,11] demonstrate the idea of decentralized approach to solve traffic problem with a knowledge-based approach. Every junction has a self-learning expert system interconnected with adjacent junctions. However, as no test data is available, performance of this approach has not been evaluated properly.

Compared with centralized control, decentralized control requires less computation power. Since the amount of data in the decentralized control is much smaller than that in the centralized one. As a result, decentralize is the most suitable approach for flow-responsive traffic control.

2.6 Junction Coordination

Vehicles move from one junction to another mostly in form of platoon. The ideal condition for drivers is that they do not need to stop when passing through junctions. That is, when a vehicle reaches a downstream junction, the signal turns green and gives go-aheads sign to the vehicles. Therefore, the traveling time from one junction to another is the optimal signal offset between these two junctions. The period that go-ahead signals coincide is called common green period.

However, there may be some vehicles queuing at the downstream junction in reality. If the signal offset is fixed, vehicles coming from the upstream junction will have to wait until the queue in the downstream is discharged completely. As a result, unnecessary delay is imposed on the vehicles. Besides, if the upstream and downstream junctions have different cycle times, the common green period will become smaller and smaller. The cycling period is the lowest common multiple of the cycle time of two junctions.

Effective signal coordination can reduce queues accumulated in junctions. Robertson and Wilson [33] used on-site experiment to compare coordinated control with isolated signal control. They proved that effective signal coordination could reduce the average queues from 4 to 10 vehicles.

Coordination of traffic signals in a road network should optimize the cycle length, phase split and offset of all signals in a road network, so that the total delay, number of stops or some other objective function may be optimized. Since a road network consists of a large number of signalized intersections, it is usually divided into a number of sub-networks to facilitate computation. Many models, like TRANSYT, were developed to determine the optimal timing plan for traffic signals within a sub-network.

The coordination of sub-networks is a complicated process. In many cases, 2 interfacing sub-networks do not have the same optimal cycle lengths. If different cycle lengths are used for adjacent sub-networks, the relative timing between the sub-networks will deviate from the initial setup. The deviation is increased by the difference in the cycle lengths in every cycle until the 2 sub-networks return to their original setups.

3 Intelligent Control

3.1 Overview

A long-standing problem in traffic engineering is to optimize the flow of vehicles through a road network. Effectively utilizing the timing of the traffic signals at intersections in the network is generally the most cost-effective means of achieving this goal. However, because of many complex aspects of a traffic system, such as human drivers' behavior and vehicle flow interactions within the network, it has been notoriously know that the determination of the optimal signal light timing is difficult. Because of the complexity of the traffic system, traffic control is considered as a classical example of non-programmed decision-making process, i.e. decision-making characterized by the lack of well-specified and analytical means for coping with a particular problem. Heuristic approach is usually more suitable in dealing with this kind of problem.

Recently, increasing number of researchers have attempted to use advanced artificial intelligent control in traffic control. Artificial intelligent control can be divided into 2 categories, self-organized and rule-based control. Self-organized control is a model-free approach. Spall and Chin [37] used the learning capability of neural network to prevent complex modeling of traffic dynamic. However, when facing with a totally unexpected traffic condition, the neural network approach may perform poorly due to a lack of precedence. Since neural network learns by finding an equation that best fits the training data, unexpected result may return for totally unknown data. In this situation, a re -training is needed. The same problem also happens in traditional model-based traffic control strategy. In general, manual overriding is required when heavy congestion occurs. It explains why human approach is preferred when dealing with totally unexpected situation. Human approach is characterized by rough approximations. Human tends to approximate the new situation from his past experience (knowledge) and figure out the appropriate actions. Rule-based (or knowledge based) control is formulated on the above rationale.

Pappis and Mamdani [30] are pioneers of applying fuzzy rule-based system in traffic control. They showed that a fuzzy logic approach reduced average delay at an intersection of two one-way roads when compared with a fixed-cycle traffic controller. Later Ho [17], Chiu and Chand [6] further extended this study and they all achieved significant results in minimizing delay in traffic junctions. Ho showed further reduction of average delay when the fuzzy rules are adapted to the current traffic conditions. Chiu and Chand extended the study into area traffic control, in which the cycle times, phases split, as well as offset, are determined dynamically. Their result shows significant reduction of average waiting time of vehicles at the junction. However, all models in these studies are so simple that only networks with no turning traffic are adopted. Besides, all studies, with the exception of Ho's, employed constant flow-rate for testing and performance evaluation that is totally different from our real world traffic condition.

Findler and Stapp [10] applied LISP [27] to optimize control of traffic signals. LISP is a programming language that specially designed to be used in expert system. A large set of rules is developed and a top-down search approach is used. The first rule matches the current conditions fires. The ordering of rules depends on the frequency of their usage. The system will fine-tune the control signals until they are adapted to current traffic conditions. However, LISP has a drawback that it does not facilitate reasoning, that is, if some unexpected situation happens, the expert system will fail to give any action.

Fuzzy logic, which provides approximated reasoning, suits the problem best. The mechanism of fuzzy logic operates similarly to human reasoning so that human expert can define, modify and update control rules for the system in symbolic and/or linguistic form. For example, a fuzzy rule can be in the form of: if this road has many cars waiting, then longer green time can be given.

3.2 Fuzzy Logic

Logical paradoxes and the Heisenberg uncertainty principle led to the development of multi-valued logic theory. In 1965, Zadeh [43] formally developed multi-valued set theory, the fuzzy set theory.

A fuzzy set F in a universe of discourse U is characterized by a membership function μ_F, which takes values in the interval [0,1] namely, $\mu_F : U \rightarrow [0,1]$. A fuzzy set F in U may be represented by a set of ordered pairs of a generic element u, fuzzy label, and its grade of membership function: $F = \{(u, \mu_F(u)) | u \in U\}$, linguistic value. When U is continuous, a fuzzy set F can be written concisely as $F = \int_U \mu_F(u)/u$. When U is discrete, a fuzzy set F is represented as $F = \sum_{i=1}^{n} \mu_F(u_i)/u_i$.

Given the fuzzy sets A, B, or U, the basic operations on A, B is:

1. The complement A of A, defined by

$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$

2. The union $A \cup B$ of A and B, defined by

$\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$

3. The intersection $A \cap B$ of A and B, defined by

$\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$

4. The Cartesian product $A \times B$ of A and B, defined by

$\mu_{A \times B}(x_A, x_B) = \min\{\mu_A(x_A), \mu_B(x_B)\} = \mu_A(x_A) \cdot \mu_B(x_B)$

Given a fuzzy relation R from U to V and a fuzzy set A on U, a fuzzy set B on V is induced, given by the compositional rule of inference $B = A \circ R$

$$\mu_B(y) = \max_x\{\min\{\mu_R(x,y), \mu_A(x)\}\}$$

3.3 Fuzzy Logic Control (FLC)

Lee [10] summarized the recent development on fuzzy logic control and presents a general methodology for constructing a fuzzy logic controller (FLC). A FLC comprises four principal components: a fuzzification interface, a knowledge base, a decision-making logic and a defuzzification interface. The fuzzification interface converts input data into fuzzy variables. The knowledge base contains control rules for the system and the decision-making logic makes use of these rules to produce fuzzy outputs. Then defuzzification interface converts them back to control actions that the system understands. Figure 3.1 shows a general structure of a fuzzy logic control.

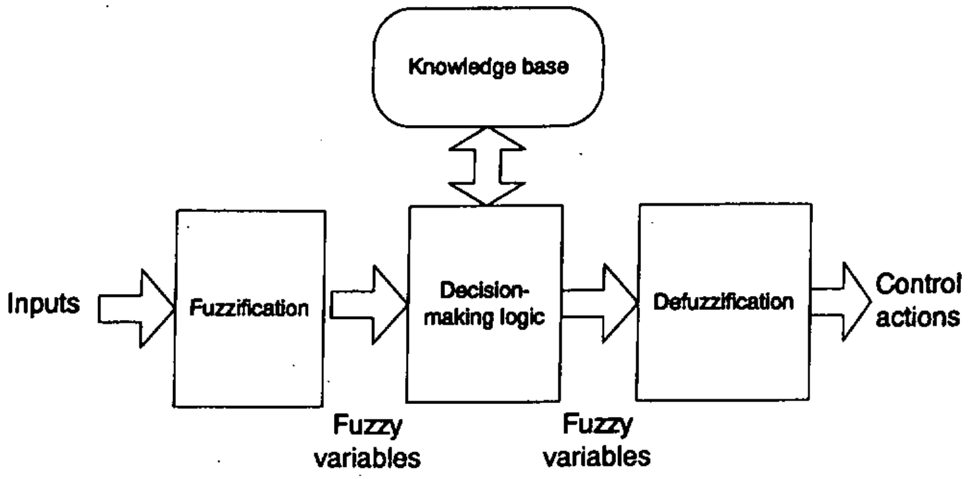

Figure 3.1 Block diagram of fuzzy logic controller.

3.3.1 Fuzzification Interface

Fuzzification includes the following functions:

- Measure the values of input variables,
- Perform a scale mapping that transfers the range of values of input variables into corresponding universes of discourse, and
- Perform fuzzification that converts discrete input data into suitable linguistic values and fuzzy labels in a fuzzy set.

Fuzzification is related to the vagueness and imprecision in a natural language. It is a subjective valuation that transforms a measurement into a valuation of a subjective value, and hence it can be defined as a mapping from an observed input space to fuzzy sets in certain input universes of discourse. Fuzzification plays an important role in dealing with uncertain information, which might be objective or subjective in nature.

In fuzzy logic control applications, the observed data are usually crisp. Since the data manipulation in an FLC is based on fuzzy set theory, fuzzification is necessary.

The choice of the mapping function, or the membership function, is based on subjective criteria of the decision. Most commonly used function is bell shaped or triangular shaped function. In particular, if the measurable data is disturbed by noise, the membership functions should be sufficiently wide to reduce the sensitivity to noise. Figure 3.2 shows a triangular shaped membership function.

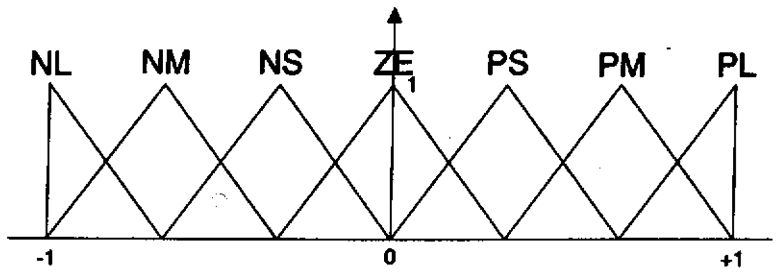

Figure 3.2 Example of triangular shaped membership function.

3.3.2 Knowledge base

A fuzzy system is characterized by a set of "if-then" rules. The collection of fuzzy control rules forms the knowledge base of a FLC. The rules always has the following form:

R_1: if x is A_1 and y is B_1 then z is C_1,

R_2: if x is A_2 and y is B_2 then z is C_2,

...

R_n: if x is A_n and y is B_n then z is C_n.

Where *x*, *y* and *z* are linguistic variables representing two process state variables and one control variable respectively. A_i, B_i and C_i are linguistic values of the linguistic variables *x*, *y* and *z* in the universes of discourse *U*, *V* and *W* respectively.

There are three modes of derivation of fuzzy control rules, as reported in [18]. They are not mutually exclusive, and it seems that a combination of these modes is an effective method for the derivation of fuzzy control rules.

1) Expert Experience and Control Engineering Knowledge

Fuzzy control rules have the form of fuzzy conditional statements that relate the state variables in the antecedent to the process control variables in the consequent. It should be noted that in our daily life most of the information on which our decisions are based on is linguistic rather than numerical in nature. Fuzzy control rules provide a natural framework for the characterization of human behavior and decisions analysis.

The formulation of fuzzy control rules can be achieved by means of two heuristic approaches. By verbalizing human expertise or by interrogation of experienced experts or operators using a carefully organized questionnaire. For optimized performance, the use of trial-and-error procedure is usually a necessity.

2) Operator's Control Actions

In many industrial man-machine control systems, the input-output relations are not known with sufficient precision to make it possible to employ classical control theory for modeling and simulation. However, a skilled human operator can control such systems quite successfully without having any quantitative models in mind. In fact, human operator employs a set of fuzzy "if-then" rules to control the process. As pointed out by Sugeno [38], to automate such processes, it is expedient to express the operator's control rules as fuzzy if-then rules employing linguistic variables. In practice, such rules can be deduced from the observation of human controller's actions in terms of the input-output operating data.

3) Self-Learning

Many FLCs have been built to emulate human decision-making behavior, but few are focused on human learning, the ability to create fuzzy control rules and to modify them based on experience. Procyk and Mamdani [31] introduced the first self-organizing controller. Self-organizing control policy is designed to change with respect to instantaneous performance measures and hence it can react to changes in the dynamics of the controlled process. It is a direct adaptive controller, which performs two tasks: generation of an appropriate online control action from the observation of the process state, and modification of the control rule-base based on a performance index. Another form of learning is by the application of an additional algorithm, Genetic Algorithms (GA), to enhance the rules. GA can be used for synthesis of fuzzy controller structures by deriving optimal sets of linguistic rules or their membership function [21,22,29].

3.3.3 Decision-making Logic

The decision-making logic is the core of the whole fuzzy logic control system. By following simple fuzzy implication function, fuzzy control actions can be obtained using fuzzy rules in the knowledge base.

A fuzzy rule in the form: "if x is A_1 and y is B_1 then z is C_1 " can be calculated by $\mu_{C_1}(z) = \min\{\mu_{A_1}(x), \mu_{B_1}(y)\}$. Fuzzy union represents the composition of multiple rules:

$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$.

3.3.4 Defuzzification Interface

Basically, defuzzification is a mapping from a space of fuzzy control actions defined over an output universe of discourse into a space of non-fuzzy control actions. At present, the commonly used strategies are the "max" criterion, the mean of maximum, and the center of area.

1) Max Criterion Method

The "max" criterion produces the point at which the possibility distribution of the control action reaches a maximum value.

2) Mean of Maximum Method

This method generates a control action, which is the mean value of all local control actions whose membership functions reach the maximum. More specifically, in the case of a discrete universe, the control action may be expressed as

$$z_0 = \sum_{j=1}^{l} \frac{w_j}{l}$$

Where w_j is the support value at which the membership function reaches the maximum value $\mu_z(w_j)$, and l is the number of such support values.

3) Center of Area Method

The widely used center of area strategy generates the center of gravity of the possibility distribution of a control action. In the case of a discrete universe, the method yields

$$z_0 = \frac{\sum_{j=1}^{n} \mu_z(w_j) \cdot w_j}{\sum_{j=1}^{n} \mu_z(w_j)}$$

Where n is the number of fuzzy labels of the output.

3.4 Genetic Algorithm (GA)

Rules definition in FLC is always a major problem for the system designers. Usually, all rules are derived from human experts and in form of linguistic knowledge. This method works well for a simple system. However, for a complex control problem, it is very time-consuming. Since the designers cannot express their knowledge easily in the form of fuzzy control rules, trial-and-error is the general approach. Researches have been conducted to investigate automatic learning methods for designing FLC, such as self-organizing fuzzy logic control and GA based learning [21,22,29], in order to derive an appropriate knowledge base for the controlled system without the necessity of its human operator.

Fusion of fuzzy systems and GA has recently attracted interests and a number of successful applications have been reported, including robot motion planning [36] and shop floor job scheduling problems [20]. GA has also been applied to fuzzy pattern classification from numerical data [19] where a minimal compact set of rules is constructed to maximize the number of correct classifications. Following their successful applications to a variety of learning and optimization problems, GA has been proposed as a learning method that can enable automatic generation of optimal parameters for fuzzy controllers.

3.4.1 Genetic Algorithm Overview

Genetic algorithms (GA) are exploratory search and optimization procedures that were devised on the principles of natural evolution and population genetics. The basic concepts of GA were developed by Holland [18] and have subsequently been extended in several research studies. Goldberg [14] provides comprehensive overview and introduction to GA.

Typically, the GA starts with little or no knowledge of the correct solution and depends entirely on responses from an interacting environment and its evolution operators to arrive at good solutions. By dealing with several independent candidates, the GA samples the search space in parallel and is hence less susceptible to getting trapped on sub-optimal solutions. In this way, GA has been shown to be capable of locating high performance areas in complex domains without experiencing difficult problems with high dimensionality or false optima, as may occur with gradient descent techniques.

The GA processes imitate natural evolution, and hence include bio-mimetic operations such as reproduction, crossover and mutation. A conventional GA has 6 features: genes, population size, fitness, selection and reproduction, crossover, and mutation.

3.4.2 Genes

A gene encodes all parameters to be optimized. The most commonly used encoding method is binary coding. A system of 2 control parameters, 1 control action with 16 levels of differences can be encoded using a 12 bits long gene.

$A_3A_2A_1A_0B_3B_2B_1B_0C_3C_2C_1C_0$

Where $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ is a binary encoded control parameter A and B respectively, and $C_3C_2C_1C_0$ is the binary encoded control action C.

3.4.3 Population Size

Population size defines the dimension of parallel searching. Larger population size allows searching of more locations at the same time, but may lead to slower convergence. The population size of two means at each generation, at most two genes can survive.

3.4.4 Fitness

Each gene is associated with a fitness value. If the performance function is to maximize the fitness value, then a higher fitness means the better chance the gene can survive.

3.4.5 Selection and Reproduction

The selection process is to choose a suitable candidate for reproduction. For maximization problem, the higher the fitness, the better chance the gene is to be selected. There are two commonly used selection processes: roulette wheel and tournament selection. In roulette wheel selection, the probability of selecting a gene is proportional to how good the gene is in this generation. Tournament selection randomly picks two genes over the population (tournament size of two), and the better one wins.

The selected genes are used for reproduction to produce offspring of the next generation. Reproduction takes place using crossover operator and mutation operator. This process results in individuals with higher fitness values obtaining one or more copies in the next generation, while low fitness individuals may have none.

3.4.6 Crossover Operator

Crossover is the most dominant operator in most GA, being responsible for producing new trial solutions. Under this operation, two genes are selected to produce new offspring by exchanging portions of their structures. The offspring may then replace weaker individuals in the population.

Parent 1: $a_1a_2a_3a_4$ $b_1b_2b_3$ b_4 $t_1t_2t_3t_4$
Parent 2: $a'_1a'_2a'_3a'_4$ $b'_1b'_2b'_3$ b'_4 $t'_1t'_2t'_3t'_4$
Positions: ^
Child 1: $a_1a_2a_3a_4$ $b_1b_2b_3$ b'_4 $t'_1t'_2t'_3t'_4$
Child 2: $a'_1a'_2a'_3a'_4$ $b'_1b'_2b'_3$ b_4 $t_1t_2t_3t_4$

Figure 3.3 Single point crossover.

3.4.7 Mutation Operator

Mutation is a bit-wised operator, which is applied with a very low probability, typically 0.1% per bit or less. If the mutation rate is set too high, it will act like cancer and produce lots of bad genes. Its role is to alter the value of a random position in a gene. Together with reproduction and crossover operators, mutation acts as an insurance against total loss of any genes in the population by its ability to introduce a gene, which may not initially have existed or was lost through application of the other operators.

```
Before:    0 0 1 1   1 0 0 0   1 1 0 1
Positions:     ^         ^
After:     0 0 0 1   1 0 1 0   1 1 0 1
```

Figure 3.4 Mutation.

3.5 Genetic Algorithm Fuzzy System

A common difficulty in fuzzy systems is the need for their parameters to be specified by a human designer. Following the successful application to a variety of learning and optimization problems, GA has been proposed as a learning method that enables automatic generation of optimal parameters for fuzzy controllers, based on objective criteria.

There are 2 application approaches to integrate fuzzy logic and GA. Firstly, linguistic rules of conventional fuzzy controller are fixed and their membership functions are optimized [22,29]. One drawback with this approach is that the ability to interpret and explain the behavior of the fuzzy controller may subsequently be lost, since the membership functions are no longer associated with any one linguistic name. Secondly, the membership functions of specified linguistic values are fixed and the GA is used to determine an optimal set of rules for the application [21]. Figure 3.5 shows general system architecture of a genetic fuzzy system.

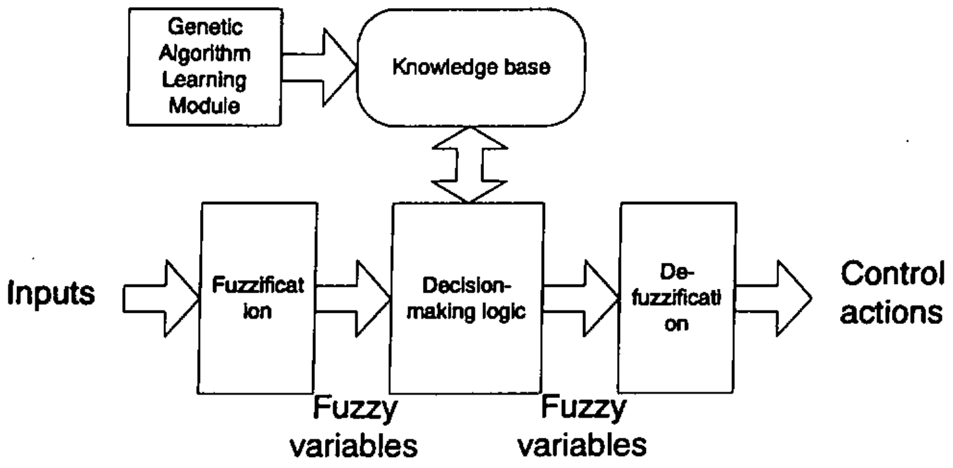

Figure 3.5 Block diagram of Genetic Fuzzy System.

4 Principle of Dynamic Programming

4.1 Introduction

Dynamic programming is a way of solving decision problems by finding an optimal strategy. The types of problems which dynamic programming is concerned are problems which can be broken down into a sequence of single decisions made one after the other and problems in which several decisions can be separated to give this same structure. The most well known example of this kind of problem is the shortest path problem.

Dynamic programming is based on the principle of optimality, which facilitates the identification of optimal policies. The principle of optimality can be formally enunciated as:

- An optimal policy has the property that whatever the state the system is in at a particular stage, and whatever the decision taken in that state, then the resulting decisions is optimal for the subsequent state. OR
- An optimal policy is made up of optimal sub policies. OR
- An optimal policy from any state is independent of how the state was achieved and comprises optimal sub policies from then on.

The principle of optimality can be proved by contradiction. If the system is in state A at some stage and the optimal policy takes it to state B at the next stage, then the optimal sub policy from B must coincide with the corresponding part of the optimal policy from A. If not, then one or the other could be improved.

The principle of optimality implies that an optimal policy is independent of the past and only looks to the future. This is an essential part of dynamic programming, as it allows policies to be calculated recursively.

Problems to which dynamic programming has been applied are usually stated in the following terms. A physical, operational, or conceptual system is considered to progress through a series of consecutive stages. At each stage the system can be described or characterized by a relatively small set of parameters called the state variables. At each stage, no matter what state the system is in, one or more decisions must be made. These decisions may depend on either stage or state or both. The decisions depend only upon the current stage and state. When a decision is made and the system undergoes a transformation to the next stage. The overall purpose of the staged process is to maximize or minimize a certain performance function of the state and decision variables, the objective function.

The key elements that a dynamic programming problem is associated with are stages, states, decisions, transformations, and returns.

4.2 Stages

The concept of a stage is required so that decisions can be ordered. Most often, the stage variable is discrete. For example, a stage can be a single phase in signal cycle.

4.3 States

A state space is a nonempty set Λ. An element $\lambda \in \Lambda$ is called a state and is a description of one of the variables (or sets of variables) that describe the condition of the system or process under study. In this study, a state is defined as queue lengths and flow-rates of all the traffic streams. An important property of states is that current and future returns depend only on the current state and not on the particular history of previous states and decisions.

4.4 Decisions

The state of the system must contain all the information that is required to identify the set of allowable decisions at some given stage. Therefore, associated with each state $\lambda \in \Lambda$ there is a nonempty set X_λ which is called the decision set for λ. An element $x_S(\lambda) \in X_\lambda$ is called a decision or decision variable and represents one of the control actions that is available when the system or process is in state λ at stage s. The decision set X_λ consists of the set of all possible control actions that might be made when the system is in state λ. In this study, a decision is the duration of green time that a phase has.

4.5 Transformations

The process under study in a dynamic programming problem evolves from stage to stage. As it does so it moves through one of the states $\lambda \in \Lambda$ to another state in Λ, as a result of the decision $x_S(\lambda)$ that is made in state l at stage s. In other words, if the process is in state l, choosing decision $x_S(\lambda)$ determines a set of states $T(\lambda, x_S(\lambda))$ to which the process can move from state λ. The set function $T(\lambda, x_S(\lambda))$ is called a transformation function and it determines the way in which the process evolves from state to state.

4.6 Policies

The return of a system depends upon the decisions that are made at each stage. A policy is a sequence of decisions, with one decision for each state.

4.7 Returns

Since a dynamic programming problem is an optimization problem, an objective function can be evaluated for some given policy. It is possible for the return functions to vary from stage to stage. The total return is a combination of the stage returns, accumulating as the process moves from state to state.

4.8 Advantages

The prime advantage of dynamic programming is that it transforms a single n-stage optimization problem into n one-stage optimization problems so that the original problem can be solved in a number of stages. Besides, the dynamic programming provides the global rather than the local optimum. Furthermore, dynamic programming is more computationally effective than complete enumeration of all feasible solutions, such as the breadth first search algorithm.

5 Fuzzy Logic Control for Single Junction

5.1 Overview

A city infrastructure constitutes of numerous interconnected junctions. Congestion in a traffic junction may spread out and affect its neighborhood. In the worst case, traffic congestion in the entire network will occur. Moreover, modern way of life relies on road network heavily. People travel from one place to another every day, for work, for school, or even for recreation. Mass transit can be one of the choices for traveling from one suburb to another. However, within a local region, road network is still the only choice. As a result, network capacity is one of the major concerns in urban planning. In newly developing area, the easiest thing is to build more roads such that it can handle very large capacity of traffic. However, this cannot apply to developed area, where land and road layout is limited. As a result, traffic control is an optimization problem on how to properly allocate the limited resource to all road users in a road network. There are two commonly used approaches, centralized and decentralized control.

Centralized control divides the entire network into many sub-networks. One central controller is used in each sub-network to monitor and apply appropriate control within the sub-network. The controller gathers information from different parts of the sub-network and computes a signal-plan for junctions in the sub-network. Since the controller has to handle a huge amount of information when computing the signal-plan, the computation power, frequency of recalculation, and calculation algorithm directly determine whether it can be used as a real-time or off-line controller. Since real-time process requires fast-decisions, the size of a sub-network cannot be too large (normally consists of few junction only) for a real-time centralized control.

On the other hand, local controllers are used in decentralized control. Local controllers are installed in each single junction. All data (local traffic information as well as coordination parameters) will be processed locally without considering the whole network at the time. As a result, less computation power is needed in the decentralized control when comparing with the centralized one. This computation model makes it suitable for on-line, real time process. Besides, Khalili and Macleod pointed out that when traffic at the junctions of an urban road network is optimized locally, a global optimization could be achieved by coordinating the junctions with respect to each other.

5.2 Area Traffic Control Methodology

The area traffic control objective is, very much similar to single junction traffic control, to minimize the number of stops per vehicles as well as the delay per vehicles among the entire network. Consider a network of 2 junctions with single direction traffic only, as shown in Figure 5.1. Suppose the traffic is coming from the right, vehicles stop at the first junction and queue up for red signal ahead. During the green signal, the queued vehicles are discharged and they travel in batch and form a vehicle platoon. The platoon travels along the road and finally reaches the second junction. If at the same time, signal of the second junction changes to green, the platoon will continue its travel without making any unnecessary stop. The drivers will encounter a "green-wave" along the road. Under this scenario, delay can be reduced significantly by aligning the green signals along the junctions. The offset of the green signals, optimum offset, is equal to the average time required to travel from one junction to another.

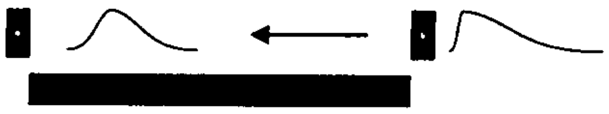
Figure 5.1 Platoon progress between 2 adjacent junctions.

Similar argument can be applied to more complex systems, but the case is not as straightforward as the above scenario. Suppose traffic from left to right is also possible, the optimal control may not be the same as before. The complexity of the problem increases if more traffic streams, such as turning traffic, are included. Optimization is needed to formulate the control of such complex system. In order to simplify the problem, priority route is usually defined first. Priority route of the junction is usually defined as the traffic stream having the highest flow. A typical example is the traffic flow from residential area into commercial area in the morning rush hours and vice versa in the evening rush hours. By giving favorable control to these directions, overall delay and stops can be reduced.

There exist two approaches in attaining the optimum offset of junction, centralized and decentralized approach. Individual junctions are grouped together in centralized control. A central controller is used to determine the offset, as well as to control signal for each junction. This approach is easy to implement and manage. The priority route will be identified first and optimization will be applied at the priority route by the central controller. However, scalability is one of the limitations of this approach. The controller may require redesign if one more junction is added into the junction group. Another limitation is on the size of the junction group. The number of junctions in a group is restricted by the computation power of the controller. More junctions imply higher demand of computation power to process the information. Moreover, between different junction groups, the centralized controller cannot make any optimization. It is because there is no information exchange between different centralized controllers. Hence the performance is unsatisfactory over the boundary region of the junction group.

Another approach is the decentralized control. This can be viewed as a special case of centralized approach in which the junction size equals to one with additional communications between adjacent controllers. It is used to coordinate between adjacent junctions. Since this approach does not put junctions together under one controller, it virtually allows unlimited number of junctions in the network. In this study, the decentralized approach is chosen.

5.3 Single Junction Controller

By using decentralized control methodology, the first criterion for area control scheme is to construct a controller capable of handling traffic at single junction independently. In this chapter, a new control method for a single junction to optimize effective green time for each phase of a cycle by fuzzy logic is presented. This controller directly updates the green time of each phase, instead of cycle time.

A number of studies have been carried out on applying fuzzy logic controller to road junction control. Pappis and Mamdani [30] provided the first study in this area. They showed that fuzzy logic reduces more delay than traditional fixed-time control does in a simple cross-junction. Chiu and Chand [6] extended Pappis's study by introducing more complex rules into the system. Ho [17] then showed that further reduction of average delay is obtained when the fuzzy rules are adapted to the current traffic conditions. However, all of them use very simple model in which there is one-way traffic only and no turning traffic is allowed. In this study, a more realistic model with both left and right turning is considered.

A number of control rules for a fuzzy logic control (FLC) system depend on the number of input and output parameters, and the number of fuzzy labels for each parameter. For example, the number of fuzzy rules for n-input, 1-output with m fuzzy labels control is m^n. The number rises exponentially when more information is taken into consideration. By dividing the single FLC into many levels, the total number of fuzzy rules can be reduced greatly. For example, by dividing a 3-input, 1-output with 6 fuzzy labels FLC into 2 levels consisting of 2-input, 1-output and 6 fuzzy labels FLC, the total number of control rules will be reduced from 216 to 72. The reduction of control rules will certainly make the implementation easier and demand less computational effort.

5.4 Traffic Model

The junction in this study is an intersection of two two-way streets with both left and right turnings. All vehicles are right-hand driven. Figure 5.2 shows all possible traffic streams that a cross-junction has.

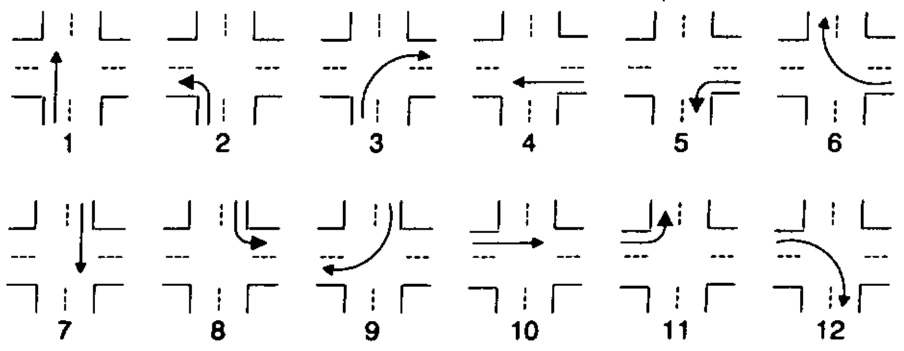

Figure 5.2 Possible traffic streams for a cross-junction with turning.

By carefully combining non-conflict traffic streams together (for example traffic streams 1, 2, 7 and 8), there are four primitive traffic streams as shown in Figure 5.3. Hence, a complete control cycle comprises 4 phases, allowing 4 traffic streams (Φ_i, i = 1,2,3,4) to pass through the junction in turn.

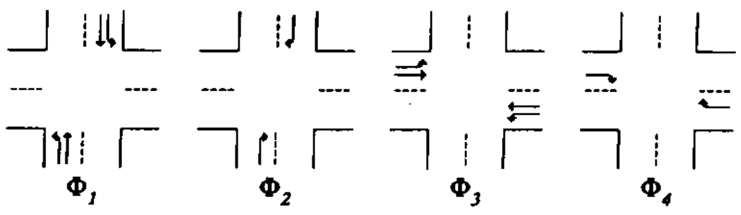

Figure 5.3 Phases setting of a junction with right turnings.

Besides, the following assumptions are made:

- All vehicles are right-hand driven, hence left turning traffic will not block the opposite traffic stream.
- Flow-rates of all traffic streams are independent of each other.
- Vehicle arrivals are uniformly distributed, with the mean equal to the flow-rate.
- Saturation flow for each arm at the junction is 1 vehicle per second. The discharge rate of vehicles (d) is the same as the saturation flow.
- Each phase has a minimum of 10s of effective green time and are keep in sequence.
- The junction can hold infinite number of vehicles at its arms.

5.5 Junction Setup

The goal of the controller is to determine the effective green times (t_g) for each phase in such a way that the total delay and queue length at the junction are minimized. To gain the real-time traffic data for the controller, vehicle detectors must be installed at the junction.

Basically, two pieces of information are most important, current queue length and flow-rate in each traffic stream. To count the number of vehicles entering the junction and to calculate the flow-rate, a vehicle detector must be installed in the vicinity of the junction. Similarly, to count the number of vehicle leaving the junction, a vehicle detector in front of the junction is needed. According to the collected data, it is possible to derive the queue length and the flow-rate.

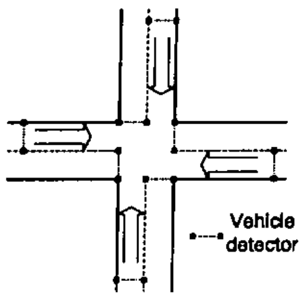

Figure 5.4 Junction setup with vehicle detectors.

Figure 5.4 shows the junction setup with vehicle detector locations. The junction-approach vehicle detectors count the number of vehicles entering the junction while the stop-line detectors count the number of vehicles leaving the junction. The stop-line vehicle detector is located in such away that it can count exactly the number of vehicles leaving in each direction. Moreover, it may act as the junction-approach vehicle detector of the adjacent junction in multi-junction configuration. Such arrangement will reduce the number of vehicle detectors installed in the network.

5.6 Hierarchical Control

According to the argument in chapter 3.1, fuzzy logic approach can handle the traffic control well. Most of the fuzzy logic traffic controllers are phase-based, so that the effective green time of the next phase is calculated. Usually, queue length and flow-rates of individual traffic stream are taken as input.

The simplest form of a FLC takes queue length as the sole input. It determines the effective green time of the next phase such that all queued vehicles can be discharged. Some preliminary experiments show the average delay and number of stops of vehicle is large. It is because every vehicle has to stop and queue before they can pass through the junction. The best way of course is to extent the effective green time such that some of the vehicles do not need to make any stop. That is the effective green time does not only discharge the queued vehicles but also those arrive after the green time commences. Therefore, the lack of arrival flow-rate makes it difficult to determine the optimum effective green time.

By adding the flow-rate as one of the input parameters, the junction controller becomes a 2-input (queue length and flow-rate) 1-ouput (effective green time) FLC. Preliminary experiments on this controller show the performance is still poor. The detail results are not shown here because they are not of primary interest. The poor performance is resulted from the fact that the controller does not know about the state of the junction. For example, when the flow-rates of all other traffic streams are small, longer time can be assigned to current phase without imposing much delay on others. As a result, the controller needs at least one more input parameter (overall junction flow-rate) in order to make optimal decision.

For a 3-input 1-ouput FLC with 6 fuzzy labels for each parameter, the maximum number of rules is $6^3 = 216$, which takes up substantial memory space and leads to substantial processing time. Therefore, it is necessary to reduce the number of rules. Hierarchical fuzzy logic control (HFLC) is therefore adopted in order to simplify the controller structure and reduce the total number of rules.

Assume that traffic stream Φ_j is given the right-of-way, the queue lengths at the traffic streams Φ_j are:

Equation 5.1 Accumulated queue of each traffic stream.

$$q_i' = \begin{cases} q_i + f_i T_{lost} + f_i t_g - dt_g & i = j \qquad (a) \\ q_i + f_i T_{lost} + f_i t_g & i \neq j \qquad (b) \end{cases}$$

Where q_i and q_i' are the queue lengths at traffic stream Φ_i before and after t_g respectively, f_i is the flow-rate of Φ_i, T_{lost} is the lost time including the short red-amber time before the green period and the time elapsed before saturation flow is reached.

From equation 5.1a, q_j and f_j can be used to determine t_g. The first level of the HFLC is a 2-input 1-output controller. This level provides an estimation of the green-time required to clear the queue accumulated just before t_g. When traffic is not saturated ($d > f_j$), from equation 5.1a, $f_j\, t_g - dt_g < 0$, q'_j will becomes zero. Therefore all vehicles on Φ_j can be discharged if t_g is large enough. Suppose $q_j' = 0$ when $t_g = t$, from (5.1a):

Equation 5.2

$$\tau = \frac{q_j + f_j T_{lost}}{d - f_j} = (q_j + f_j T_{lost}) + \frac{(q_j + f_j T_{lost})(1 - (d - f_j))}{d - f_j}$$

If $q_j + f_j T_{lost} \geq 1$, i.e. there is at least one vehicle at Φ_j initially, equation 5.2 becomes:

Equation 5.3

$$\tau \geq (q_j + f_j T_{lost}) + \left(\frac{1}{d - f_j} - 1\right)$$

As a result, t defines the maximum effective green time allocated to a phase. It should be noted that if $q_j + f_j T_{lost} \geq 1$, minimum effective green time of 10s will be given without evoking the controller.

On the other hand, t_g is also restricted by the other traffic streams. The traffic conditions in other streams are also necessary in determining t_g, which is related to the total queue length on Φ_i's: $i \neq j$ according to equation 5.1b:

Equation 5.4 Overall accumulated queue in a junction.

$$\sum_{\substack{i=1 \\ i \neq j}}^{4} q_i' = \sum_{\substack{i=1 \\ i \neq j}}^{4} q_i + T_{lost} \sum_{\substack{i=1 \\ i \neq j}}^{4} f_i + t_g \sum_{\substack{i=1 \\ i \neq j}}^{4} f_i$$

In order to consider the traffic conditions of other streams, t_g attained from the first level needs to be adjusted. The second level of the HFLC determines such adjustment according to the relation of flow-rates (i.e.: f_j and $\sum_{i=1}^{4} f_i$). The structure of the HFLC is illustrated in Figure 5.5.

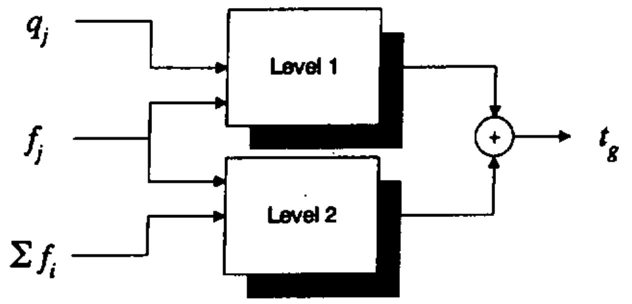

Figure 5.5 Block diagram of HFLC.

5.6.1 Level 1 – Local Traffic Stream Optimization

This level of the control is based on the equation 5.3. However, because of the inaccuracy of the vehicle detectors and highly fluctuated flow-rates, τ obtained by equation 5.3 may not be optimal, especially when flow-rates are high. Fuzzy logic, which allows uncertainty and imprecision, provides a solution. Since only a rough estimation of t_g is needed, the fuzzy rules of this level can be derived easily by a human expert.

From equation 5.3, if $f_j \le 0.6$ then:

$$\frac{1}{d - f_j} - 1 \le \frac{1}{1 - 0.6} - 1 = 1.5$$

Equation 5.5

$$\therefore \tau \ge (q_j + f_j T_{lost}) + 1.5 \approx (q_j + f_j T_{lost})$$

If the maximum traffic stream capacity is set to 60% of the saturation flow, equation

5.3 can be simplified to equation 5.5. By substituting $x = q_j + f_jT_{lost}$, the controller can be simplified as single-input (x), single-output (τ) FLC. Rules and membership functions for fuzzification and defuzzification of the first level of the controller are shown in Table 5.1 and Figure 5.6 respectively.

Table 5.1 Rules table for the first level of the Controller.

[illegible]	[illegible]
VS	VS
S	S
M	M
L	L
VL	VL
UL	UL

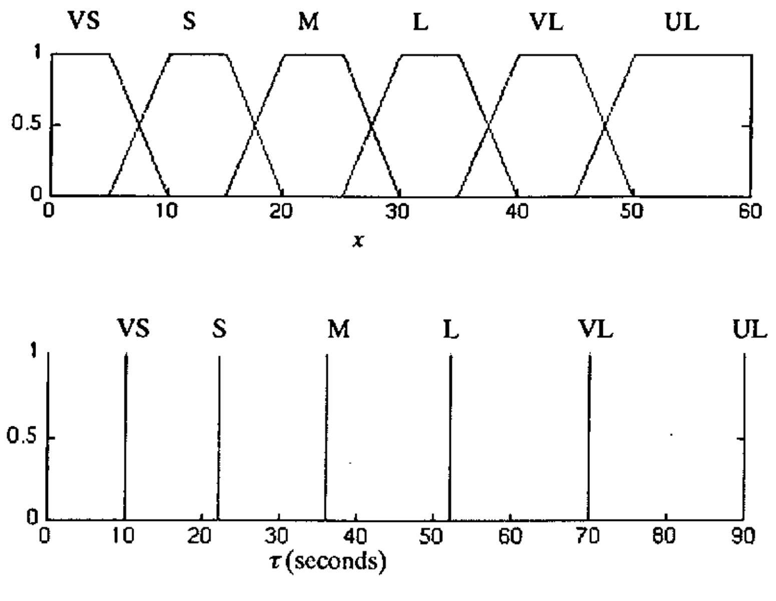

Figure 5.6 Membership functions for the first level of the HFLC.

5.6.2 Level 2 – Traffic Streams Coordination

The purpose of this level is to coordinate between different traffic streams at the junction. This level of the HFLC takes flow rate and total flow rate as input parameters. The output of this level, Δt, will fine tune τ obtained at the first level. Their sum is the length of the green time period for the following phase of a control cycle. The second level control is also a two-input (f_j and $\sum_{i=1}^{4} f_i$), single-output (Δt) FLC. The operation of the second level of HFLC is based on the equation 5.4. Since well-defined analytical definition between the flow-rates of all traffic streams is not available, the rule set on this level of control is purely from practical experience and human heuristic. The rules and membership functions for the second level are shown in Table 5.2 and Figure 5.7 respectively.

Table 5.2 Rules table for second level, flow time adjustment, of the fuzzy controller.

		Total flow rate					
		VS	S	M	L	VL	UL
Flow Rate	VS	NS	NS	NM	NM	NL	NL
	S	NE	NE	NS	NS	NS	NM
	M	NE	NE	NE	NS	NS	NS
	L	NE	NE	NE	NE	NS	NS
	VL	PS	PS	PS	PM	PM	PL
	UL	PS	PS	PS	PM	PL	PL

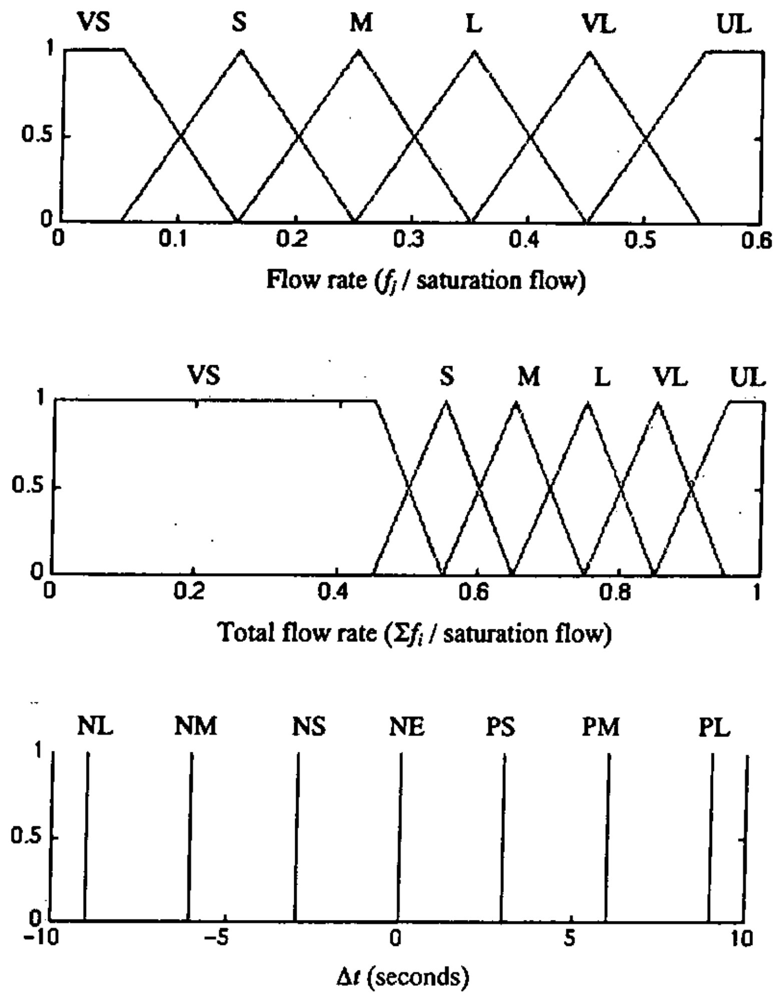

Figure 5.7 Membership functions for the second level of the HFLC.

6 Implementation of HFLC

6.1 Overview

The performance of the HFLC is evaluated by computer simulation. The simulation environment, including the hierarchical fuzzy logic controller and junction modeling, is constructed by an object-oriented approach. There are a total of 3 modules in the system, namely flow-generation module, junction module, and control module. The flow-generation module generates traffic profile for the system. The junction module simulates the environment and operation of a road junction. The control module consists of decision-making logic to determine effective green times for the junction.

6.2 Program Architecture

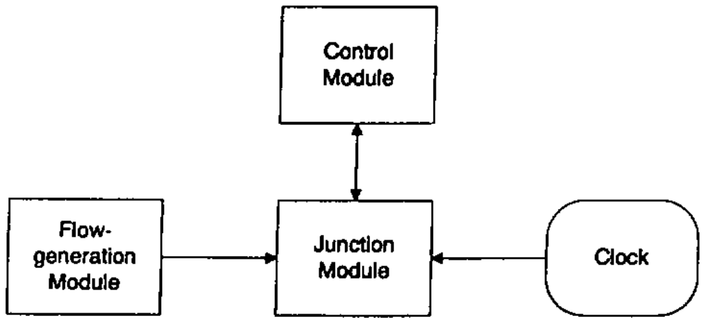

Figure 6.1 Simulation module structure.

Figure 6.1 shows the structure and interactions of the modules. When the simulation program starts, traffic profile filename can be given, indicating to the program to use a specific profile instead of generating a new one. This is useful to compare the performance of the HFLC with the fixed-time control by using the same traffic profile.

The flow-generation module reads the specific traffic profile or generates a new one. The traffic profile consists of arrival information of every traffic stream of the system. Each row of the profile represents the arrival in one second and one bit in that row stands for one traffic stream. A zero in bit 0 of the row means no vehicle arrives in traffic stream 0 in that second. Similarly, A one in bit 0 means a vehicle present in traffic stream 0. Obviously, the saturation flow will be 1 veh/s and is the same as the assumption made in the last chapter. In every second, the system will retrieve a row from the profile and create a vehicle object in specific traffic stream when necessary.

A vehicle object represents a single vehicle that exists in the system. When a vehicle enters the system, a vehicle object is created and the time of arrival is recorded. Whenever a vehicle leaves the system, its delay is calculated by equation 6.1. The vehicle object will then be destroyed to reduce memory space demand. In every 10 seconds, the average delay of the system is calculated and is used as performance indicator. Figure 6.2 shows flow-chart of the system.

Equation 6.1 Delay calculation

$$\text{delay} = \text{departure time} - \text{arrival time} - \text{normal cruising time}$$

Where normal cruising time is the journey time of the vehicle through the junction when there are no stops and delay.

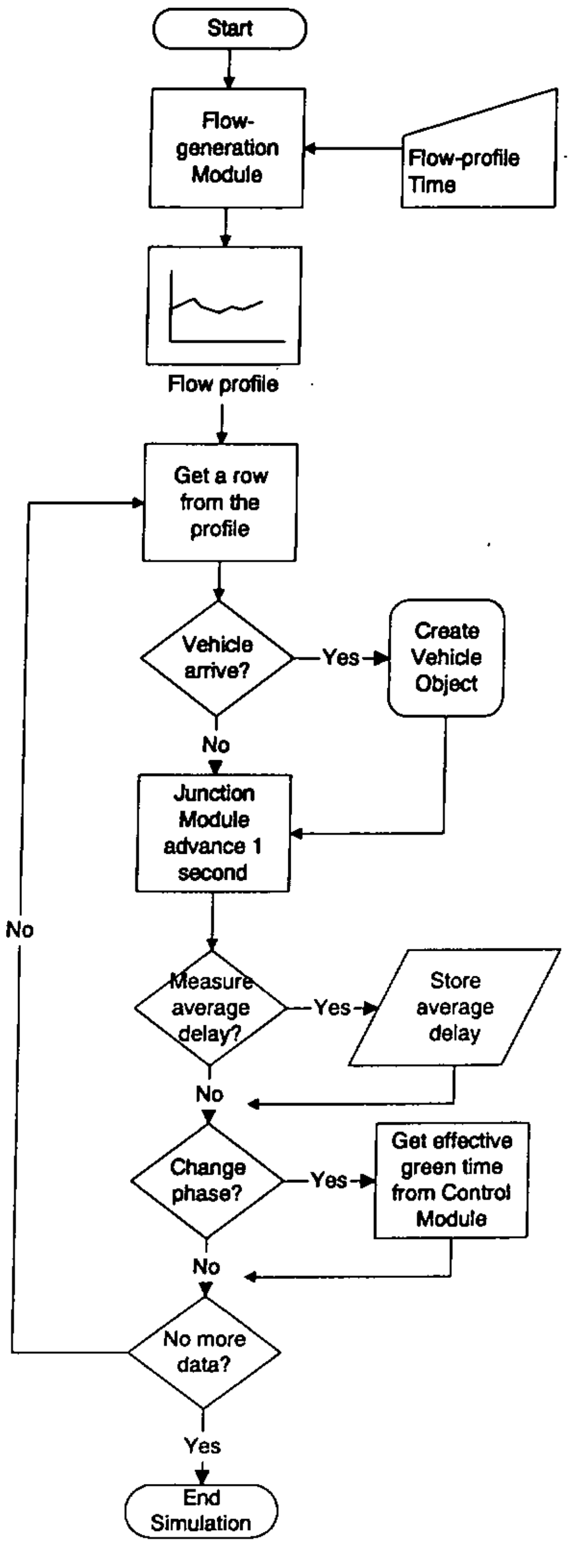

Figure 6.2 Flow-chart of the simulation system.

6.3 Flow-generation Module

The flow-generation module is responsible for the generation of traffic profile or reading the profile from a given file. It consists of 2 objects, flow-rate generation object and random number generator object. Figure 6.3 shows the object interaction of the flow-generation module.

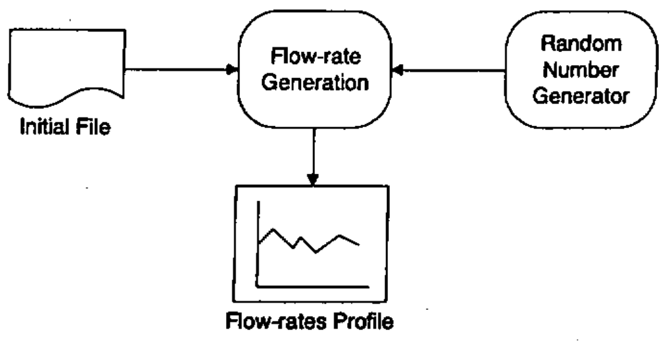

Figure 6.3 Flow-generation module structure.

The flow-rate generation object reads flow-rate from an initial file, which defines flow-rate of each traffic stream of the junction every minute. Vehicle arrives at the junction randomly with its probability equal to the flow-rate. The random number generator generates a number between 0 and 1 for each simulated second. A vehicle is present if the number is greater than the flow-rate. After one minute, the system will read a new flow-rate from the initial file.

Each row of the flow-rate profile represents vehicle arrives within one second. Each bit of the row represents one single traffic stream. For example, the first bit represents traffic stream 1 while the second bit represents traffic stream 2. The bit value of 1 indicates a vehicle is arrived.

For each traffic stream, at most 1 vehicle will arrive per second. As a result, the maximum traffic flow (saturation flow-rate) of each traffic stream is 1 veh/sec, which is consistent to the assumption on the junction in chapter 5.

6.4 Junction Module

The junction module simulates the environment and operation of a road junction. It consists of 3 different objects, vehicle object, traffic stream object, and junction object. The input and output of the junction module are the flow-rate profiles of each traffic stream and the average delay of the system respectively. Figure 6.4 shows the structure as well as object interaction within the junction module.

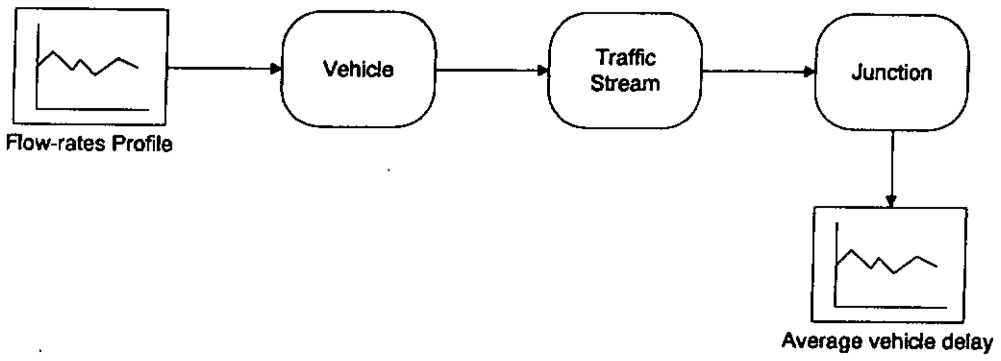

Figure 6.4 Junction module structure.

A counter in the simulation program controls the operation of the junction module. At the beginning of each second, the module will fetch a row from the flow-rates profile. This row indicates arrival information of each traffic stream. When a new vehicle arrives, a vehicle object is created and the time of arrival is marked. This object is then moved into the traffic stream object. A traffic stream object is basically a first-in first-out queue of vehicle objects. All vehicles in the queue should stop in front of the junction unless they are given the right-of-way. When a vehicle object leaves the junction, it will be destroyed after the delay of that vehicle is recorded.

For every 10 seconds, the junction module will calculate the average delay of vehicle in the system. The junction object will ask every traffic stream object to determine the overall delay and the number of vehicle currently in that stream. Then the junction will calculate the average delay of vehicle using equation 6.2.

Equation 6.2 Average delay calculation

$$\overline{D} = \frac{\sum_{i=1}^{n} D_i N_i}{\sum_{i=1}^{n} N_i}$$

Where $\overline{D}$ is the average delay of vehicle, D_i and N_i are the total delay suffered in traffic stream i and the total number of vehicles in that traffic stream respectively, and n is the number of traffic stream in the junction.

At the end of each signal phase, the junction object will ask the control module to determine the effective green time of the following phase. At the same time, a common red period of 3 seconds is given to all traffic streams. It should be enough for an ordinary computer to determine the effective green time using the HFLC because the operation is basically a table looking-up process, together with fuzzification and defuzzification operations. If the control module fails to calculate the effective green time within the common red period, the effective green time from pervious cycle of that particular phase will be used. Since the control module is time critical, it is not possible for the controller to keep waiting. As a result, the control module is a real time system, whenever the calculation fails, result from the last cycle will be used instead.

6.5 Control Module

The control module is a real-time control system, in which if the calculation fails to complete within certain time, the results will be degraded. Figure 6.5 shows the control module structure of HFLC.

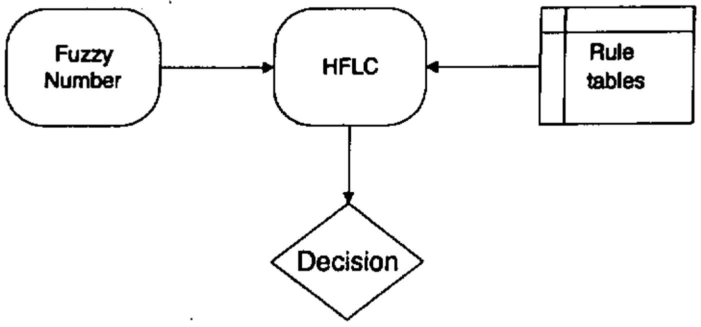

Figure 6.5 Control module structure of HFLC.

The HFLC control module consists of the rule tables and 2 objects, HFLC and fuzzy number object. The rule tables, in the form of array of integers, holds all the fuzzy rules of both level 1 and 2 HFLC control. The numbers in the array represents the fuzzy labels of the system. For example, fuzzy label "VS" of the flow-rate is represented by a value of 0. Each "if-then" rule is an element on the array. Since the system does not allow two rules that take the same input but produce contradictory output, a two-dimensional array is enough to hold all rules for a 2-input 1-output system. For example, a rule in the form "if flow-rate is VS and total flow-rate is S then Δt is NS" is located in the first row, second column of the array. The fuzzy number object is an object-oriented representation of a fuzzy value, while the HFLC object encapsulates all the operations of the controller.

When a decision is to be made, the HFLC control module will ask for the inputs from junction. The inputs are the queue length and flow-rate of the traffic streams. They will then be transformed to the fuzzy domain and is stored in the fuzzy number objects. The HFLC object lookup the rule tables and determines the results of both levels 1 and 2 of the controller. The results will then be defuzzified and transformed back to time domain. The result from both levels are added up together and become the effective green time of the next phase. If the decision fails to complete within the common red period, the effective green time of last cycle is returned.

6.6 Simulation Method

Simulation has been carried out to evaluate the performance of the HFLC. The simulation can be divided into 2 main categories, constant and time-varying traffic conditions. Constant traffic condition means that the flow-rates and turning percentage are kept unchanged throughout the simulation. It is used to test the general performance of controller over a long period of time. Time varying traffic condition means that the flow-rates, as well as the turning percentage, of vehicles vary with time. The traffic situation is time varying in reality. As a result, performance and response of the controller under real traffic conditions can be investigated. Moreover, traffic profiles for the time-varying condition are constructed based on real traffic data in Hong Kong [4].

Simulation results of the HFLC against the fixed-time control are discussed in the chapter 10. Remarkable reduction of average delay of vehicle is observed under both constant and time-varying conditions. However, in the transient period where the rate of change of flow-rate is large, the performance of the fixed-time controller is better. This indicates that the rule set for the second level of HFLC, which is currently derived by human trial -and-error technique, is not optimal. A more systematic way in deriving rules for the second level is needed. Genetic algorithms (GA), a heuristic multidimensional search algorithm, will be used to derive an optimal set of rules for the second level of HFLC. The implementation of the GA learning will be discussed in the next chapter.

7 Genetic Algorithms Optimization

7.1 Overview

Rules definition in fuzzy logic control (FLC) is always a major problem for the system designers. Usually, all rules are derived from human experts and in form of linguistic knowledge. This method works well for a simple system. For complex system, deriving rules is very time-consuming because the structure and the number of rules usually increase with the system complexity. Since the designers cannot express their knowledge easily in terms of fuzzy control rule, trial-and-error is the general approach. Research studies have been undertaken to investigate automatic learning methods for the design of FLC by deriving an appropriate knowledge base for the controlled system without necessity of its human operator.

Self-organizing fuzzy logic controller [5,25] is a hybrid system, which combines the self-organizing characteristic neural network and the fuzzy inference mechanism. This system has a fuzzy interface with the neural network inside to encapsulate all the fuzzy "if-then" rules. One major drawback of the system is that human expert cannot analyze and modify the fuzzy rules of the system. Genetic algorithms (GA) [18,14], a universal optimization and search algorithm, is employed because adaptive control, learning and self-organization can be considered as optimization or search problem in a number of cases. GA has been used for optimization of the membership functions [22], determination of optimal set of rules [21], or the combination of them [40]. Tabu search [12,13], advanced heuristic searching technique that avoids entrapment in local minimum, can also be used in refining the FLC. One major difference between the Tabu search and GA is that it remembers all the visited position within a number of generations by Tabu list. Unlike GA, the Tabu search does not waste time to search the visited position again. Thus it provides faster search. However, if the searching criteria are changing, such as time-varying flow-rate in the traffic control problem, previously visited position may become a solution under current condition. As a result, the Tabu list may become an obstacle and hence GA is preferred in the study.

7.2 GA Learning

In the computer simulation carried out in chapter 9, the HFLC generally performs better than the ordinary fixed-time one under both constant and time-varying traffic conditions. However, in the transient period of the time-varying traffic environment where the rate of change of flow-rates is large, the HFLC does not perform as well as it does otherwise. It is because the controller does not provide optimal control under all conditions, especially under time varying traffic condition.

The second level of the HFLC handles the time-varying flow-rate and the coordination among different traffic streams within the same junction. Because of the lack of well-defined analytical relationship among the traffic conditions at all streams, the rules in this level are derived primarily by trial-and-error method. The poor performance over the transient period reflects that the rules for the second level of HFLC are not optimal. Efficient method to obtain an optimal set of rules for this level is needed.

Besides, it has been assumed that the flow-rate of individual link is not greater than 60% of saturation flow. It is because all fuzzy rules in the second level of HFLC are developed using trial-and-error method and the number of rules is related to the number of fuzzy labels in each input parameter. If the number of fuzzy labels increases, the size of the rule set will increase exponentially. For example, if the number of fuzzy label increase from 6 to 16 in a 2-input 1-output fuzzy system, the size of rule table will increase from 36 to 256. To define 256 rules is too much for the trial-and-error approach. As a result, the flow-rate of individual link is limited to 60% of saturation flow maximum in order to reduce the number of fuzzy labels to 6. Obviously, this will lead to poor performance when flow-rate is greater than 60% of saturation flow. To overcome the problem, the flow-rate constraint is relaxed by introducing more fuzzy labels into the system. As a result, the trial-and-error approach is no longer appropriate and hence a systematical way of rule generation is needed.

In general, there exists 2 ways to obtain rules automatically for fuzzy logic controller, self-organization or searching. Procyk and Mamdani [31] introduce self-organizing approach as an extension to the static fuzzy controller. It has subsequently been extended in several other studies including Daley and Gill [8], Linkens and Abbod [26]. The self-organizing approach is designed to change with respect to instantaneous performance measures and hence it can react to changes in the dynamics of the controlled process. It is a direct adaptive controller that performs two tasks: generation of an appropriate online control action from the observation of the process state, and modification of the control rules based on a performance index. The structure of the self-organizing controller consists of 2 rule-base, "meta-rule" and "control rule". The "meta-rule" consists of a small set of rules and is responsible to generate control rules for the "control rule". For the derivation of the "meta-rule", human trial-and-error approach is always needed. The approach reduces the trial-and-error rule derivation from a large and complex "control rule" to a small "meta-rule" rule-based. GA is widely used in automatic fuzzy rules generation. Not only because GA is one of the most advanced searching algorithm, the capability of drifting away from local optimum and high dimensional search draws much research attention recently. By using GA searching the trial-and-error rules definition process can be removed completely. GA was applied to optimize the shape of membership functions [29,22] and to determine an optimal set of rules for the application [21,23]. However, a major drawback of using GA is that the algorithm does not guarantee to stop within a given time. As a result, GA is not suitable for real time process.

In this chapter, GA determines the rules in the second level of HFLC and those at the first level are based on a mathematical model. No human knowledge is needed in the rule definition phase. The constraint of maximum individual link flow-rate is relaxed.

7.3 Revised HFLC

The overall structure of the HFLC remains the same. It still consists of 2 levels of FLC, with 16 fuzzy labels for all inputs. Figure 7.1 shows the block diagram of the revised HFLC.

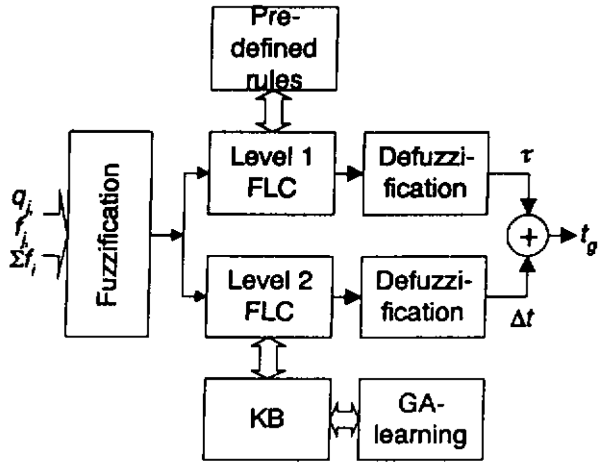

Figure 7.1 Block diagram of HFLC with GA Optimization.

7.3.1 Level 1 of HFLC

Since the constraint in maximum flow-rate of individual traffic stream is relaxed, the approximation in equation 5.5 can no longer be applied. Referring to equation 5.3, 2 inputs (q_j, f_j), instead of approximated 1 input (queue' = $q_j + f_jT_{lost}$) is needed. As a result, the first level of HFLC will become a 2-input 1-output fuzzy logic controller. The revised flow-rate membership functions and control surface for this level are shown in Figure 7.2 and Figure 7.3 respectively.

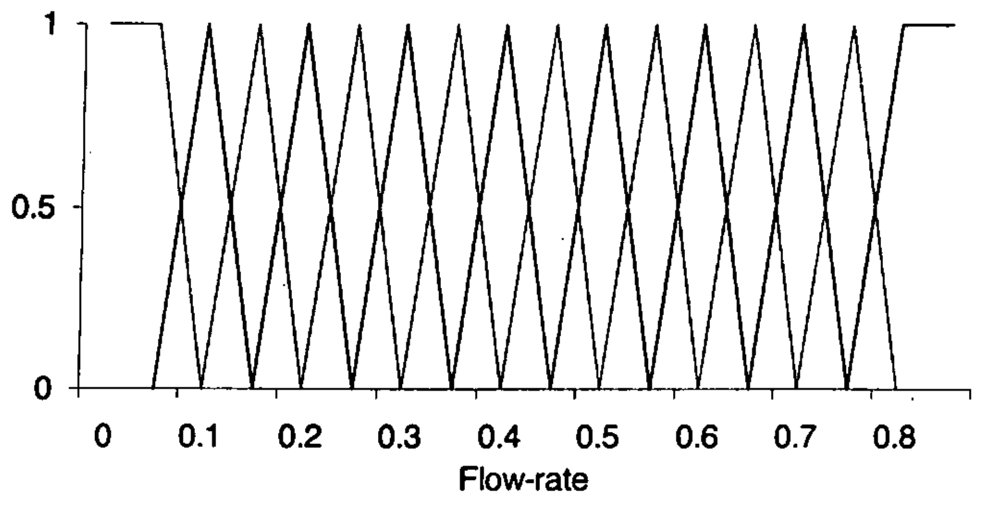

Figure 7.2 Flow-rate membership functions for the first level of the HFLC.

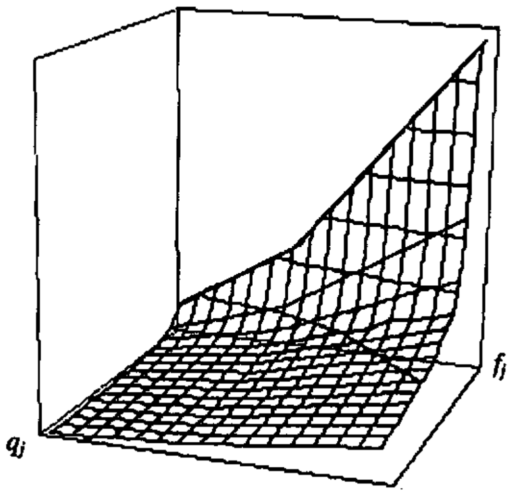

Figure 7.3 Control surface for the first level of the HFLC.

7.3.2 Level 2 of HFLC

The structure of the second level of HFLC remains very much the same. Similar to the first level, the second level is also a 2-input 1-output FLC with 16 fuzzy labels for each input parameter. As a result, there will be 256 rules in this level too.

Referring to the chapter 5.6, the second level of HFLC is responsible for dynamic properties of the flow-rate as well as the coordination between different traffic streams within the junction. Since well-defined analytical relationship between the flow-rates of all traffic streams is not available, the rule set on this level of control is purely from practical experience and human heuristic. A systemically and automatic searching algorithm, GA, is used to replace the rule definition phase at this level of control.

When the training starts, a set of training data containing combinations of different flow-rate is used. The data contains both constant flow-rates over a period of time and periodically time-varying flow-rates. The GA learning module as shown in Figure 7.1 will randomly produce candidate, fuzzy rule, for the second level control. Simulation is performed to test if the candidate is suitable or not. Suitable candidate will be added to the system. The training will terminate if no candidate is being promoted within a number of generations. Since the time needed for GA to finish searching is not known, the training process can only be performed offline. Detail training process is discussed in the next chapter.

7.4 Concluding remarks

The hierarchical controller encourages the use of automatic searching by significantly reducing the size of the search space. For a 2-input 1-output system, the search space size is N^2, where N is the number of fuzzy labels used by each input parameter. Nevertheless, the search space for a 3-input 1-ouput system is N^3. The hierarchical form divides the 3-input 1-output FLC into 2 levels, with a 2-input 1-output FLC at each level and only the second level needs automatic searching to establish the rule-set. As a result, the HFLC reduces the size of search space from N^3 to N^2.

GA is used to automatically search the optimal set of rules for the second level of the HFLC. Without the requirement of initial set of rules, GA is more suitable in system tuning than the self-organizing approach. Moreover, the GA searching is an adaptive process that can be started at any time if the control environment changes. Unlike the neural-fuzzy hybrid system, the GA-fuzzy does not require the knowledge of the changes in controller structure. As a result, the GA searching can be applied to any kind of fuzzy system very easily.

8 Implementation of GA Learning System

8.1 Overview

A Genetic Algorithms (GA) based offline learning system is constructed in this chapter. It is used to obtain fuzzy rules in the second level of HFLC automatically without any human interaction. The GA will search the entire search space, generate or remove any contradicting rules. As a result, the GA is capable of producing a small set of rules for the system.

8.2 GA Learning System

Each gene encodes a fuzzy 2-input 1-output fuzzy if-then rule. The encoding and decoding process of a gene are described later in this chapter. The system has 2 tables, KB and limbo. The KB is the knowledge base on the second level of HFLC and contains all necessary rules for this level. Initially, the KB is empty. A gene is promoted into the KB by the learning process if it is found suitable. Gene retained in the KB remains in the system unless the GA learning system finds a contradicting gene. It is to ensure that the KB contains all the necessary rules useful to the system. The limbo is a temporary placement of newly generated gene and can store up to 20 genes. The size of the limbo determines the number of genes which can be searched at a time. It does not affect the quality of the result very much. However, if the size is too small, pre-mature convergence will occurs. If setting the size too large, the algorithm will spend more time to converge.

In order to determine the size of the limbo, simulations under carefully controlled environment are preformed. The vehicle arrival pattern and queue length of every traffic streams are fixed in each generation. This is to ensure that only 1 rule will give out the best fitness and this rule will be the same throughout the entire simulation. The testing was repeated 20 times and the percentages of producing the correct rule are recorded. Table 8.1 shows the successfulness of the learning against the limbo size.

Table 8.1 Percentage in producing correct rule under different limbo size.

Limbo Size	Percentage in producing the desired rule
2	0 %
5	5 %
10	65 %
20	85 %
30	80 %
40	90 %
100	80 %
200	85 %
500	90 %

In the above table, by increasing the limbo size will have much higher chance of obtaining the correct rule. Starting from the limbo size of 10, the correctness percentage is acceptable. However, because the gene reproduction is based on probability, there is no guarantee in producing the correct rule at each simulation. As a result, none of the test cases produce 100% successful rate.

The limbo size with reasonable correct rule production percentage, 20, is chosen. It is because the actual rule quality cannot be guaranteed at each learning process. If the process can only generate the near-the best rule this time, that rule will be removed if a better rule is found later.

During the learning process, a gene in the limbo will be tested together with the KB. If the gene is proved "good" to the system, it will be promoted into the KB. A "good" gene means it produces less delay when it is used. If the promoted gene contradicts to other genes in the KB, both the promoted and the contradicted genes will be removed.

Selecting and mating genes in the limbo produce new gene. Genes are selected according to a fitness value associated with the gene. Moreover, old gene will be removed after certain generations if it is not promoted to the KB. The process continues until no new gene is being promoted within 100 generations or 5,000 generations reached. This 2 figures are used to act as a save exit conditions in order to prevent the program computes infinitely.

8.3 Gene Structure

A gene contains all parameters required for using genetic algorithms. In the HFLC, fuzzy rules in the second level are target of optimization. A rule has the form:

If f_j is A_j and $\sum f_i$ is B_i then Δt is T_k.

Where *A, B, T* are fuzzy values and each has 16 possible fuzzy labels.

A gene is divided into 3 parts, 2 for input parameters and 1 for the output parameter. Moreover, since all parameters can have 16 different fuzzy labels, a 4-bit wide binary string ($2^4 = 16$) is enough to encode one parameter. As a result, the minimum length of a gene to encode the entire fuzzy rule will be 12 bits.

In practice, a computer will fetch a word instead of individual bit from memory at one time. A word can be 16 bits or 32 bits wide depending on hardware implementation of a computer. Moreover, the smallest processing unit of a computer is byte, which is 8 bits wide binary string. Any data representation inside a computer should be a multiple of 8-bit wide (8, 16, 32, etc). As a result, a gene will be 16 bits (2 bytes) wide, with the first 4 bits left unused and set to all 0. Figure 8.1 shows the structure of a gene.

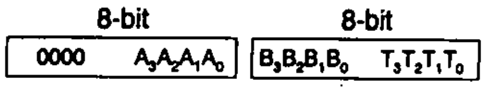

Figure 8.1 Structure of a Gene.

Both the crossover and mutation operations start from the fifth bit (A_3) of a gene. As a result, all operations do not affect the first 4 bits and hence no extra checking operation needed to ensure gene produced is valid.

Besides, every gene in the limbo is associated with 3 variables, "fitness", "gene age", and "number of fires". Fitness is calculated by equation 8.1 and is a performance indicator of how good the gene is in the system. The higher the fitness, the better the gene is for the system. The objectives of the learning algorithm are to minimize average delay $\overline{D}$ per vehicle in all traffic streams (resources allocation) and to maximize the number of vehicles C_j leaving the junction (resources utilization) within a signal-phase. Thus the fitness function F is defined and the GA aims to maximize it.

Figure 8.2 GA fitness function.

$$F = C_j - \overline{D}$$

"Gene age" represents the age of a gene. If a gene is too old ("gene age" > 20), it will be removed from the limbo. The age is related to the number of generations a gene survives. When one generation passes, the age of the gene will be increased by one. Only 5000 generations is chosen as the termination criteria of the learning system. The system is allowed to test a number of genes before stopping it. "Number of fires" indicates how active a gene is. In every generation, the system will generate a vehicle arrival pattern by using defined flow-rates. Simulation will then be performed on the KB alone and the fitness is recorded. Each gene in the limbo will be selected one at a time and the system uses the same arrival pattern to test the candidate together with the KB. The candidate is set fired when it has higher fitness than the KB. Once a gene is set fired, its "number of fires" will be incremented by 1. If the "number of fires" of a gene is greater than 10, which is the firing percentage ($\frac{\text{number of fires}}{\text{gene age}} \times 100\%$) is greater than 50%, the gene will be promoted to the KB. The percentage is set in such a way that only those highly active genes will be promoted and hence the optimal set of rules will be produced.

8.4 Learning Algorithm

The GA allows learning with both constant and time-varying flow-rates. A set of flow-rate combination is defined to cover all possibilities up to saturation. The traffic condition under each combination is fed to the system for training in turn. In other words, the rules are trained under a wide range of traffic conditions.

All genes (or rules) generated by GA are placed in limbo first. Initially, the KB is empty. An arrival profile, based on the predefined flow-rate combinations, is generated. At the beginning of each phase of a red-green cycle, an effective green time t_g for the current phase is first calculated using the HFLC. The traffic at the junction is then simulated until the end of this phase according to the calculated arrival profile and the final queue lengths of the pervious phase. The queue lengths at the end of one phase are the initial queue lengths of the next one. One gene from the limbo is added into the KB to calculate a new effective green time t_{gi}. The traffic is simulated again using t_{gi} with the same arrival profile. The average delay of vehicles at the junction and the number of vehicles leaving the junction are recorded for each gene in order to evaluate the performance of the controller with respect to its fitness value. The fitness function will be discussed in the next section. The basic idea of GA learning is to maximize the fitness value such that suitable genes (or rules) can be produced.

Any gene from the limbo will be set "fired" only if it achieves better fitness value than when it is not used. A gene from limbo is promoted to the KB if it is very active ("number of fire" > maximum number of fire, 10). If a gene in the limbo is very old ("gene age" > maximum gene age, 20), it will be deleted from the limbo. The numbers are just picking randomly, but they cannot be too large. The maximum gene age indicates how long a gene can residue in the limbo. Since the limbo size and the maximum generation are small, a gene cannot residue in the limbo for a long time. As a result, a small number is set. The evaluation carried out in table 8.1 is based on the values described above.

Genes in the KB will not be removed unless they contradict with others. The genes are said to be contradicting only if they produce completely different output when they have the same input.

Each generation is represented by a signal-phase. After 1 signal-phase, GA selection and reproduction will take place. At each generation, 10 genes will be selected. The selection criterion is based on the fitness value associated with the gene. Genes with higher fitness value will have better chance of being selected. Roulette wheel selection, despite its slower convergence rate, instead of tournament selection is used so that the algorithm can examine as many rules as possible. The 6 selected genes will be randomly grouped into 3 pairs. Single point crossover will be applied to those gene pairs and the mutation rate of the algorithm is set to 0.1%. The newly generated genes will be placed into the limbo. When there are more than 20 genes in the limbo, genes with lower fitness value will be removed.

The process continues until no gene is promoted into the KB within 100 generations or the maximum generation of 5000 is reached. Figure 8.2 shows flow-chart of the GA learning.

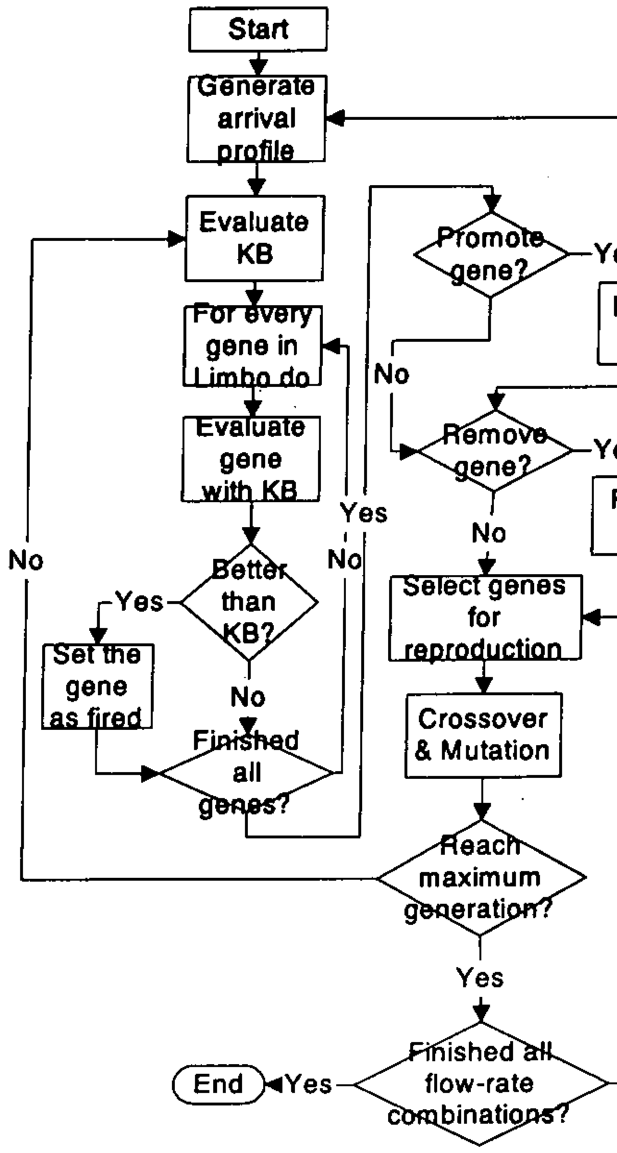

Figure 8.3 Flow chart of the GA learning process.

8.5 Execution Time Reduction

The learning time for the GA depends on the time required to evaluate a gene. For example, if a gene needs 0.1s in 1 generation, a limbo size of 20 genes will take 2 seconds in 1 generation and 1000 generations will need 33 minutes 20 seconds to complete. The extensively long evaluation time of a single gene is the time taken to simulate a whole signal-phase before the fitness value can be calculated.

Since every gene will be tested using the same vehicle arrival pattern, carefully rearranging the process can shorten the time needed for the GA learning. The output of all genes is the effective green time of the signal-phase. To reduce the simulation time, output of all genes will be determined first. A list of effective green time will be produced and sorted in ascending order. When the simulation start, the effective green time is set to the longest time produced. If gene 1 together with the KB produces t_1 seconds of effective green time, while gene 2 produces t_2 seconds, after simulating t_1 seconds, the result will be the fitness value of gene 1 and the result after t_2 seconds will be the fitness of gene 2. Hence, only 1 simulation is needed rather than 20 will be needed to produce all the fitness values. Figure 8.3 shows space-time diagram of the enhanced learning method. As a result, for a limbo size of 20, the revised method will be around 20 times faster than the original one and 100 seconds is needed only.

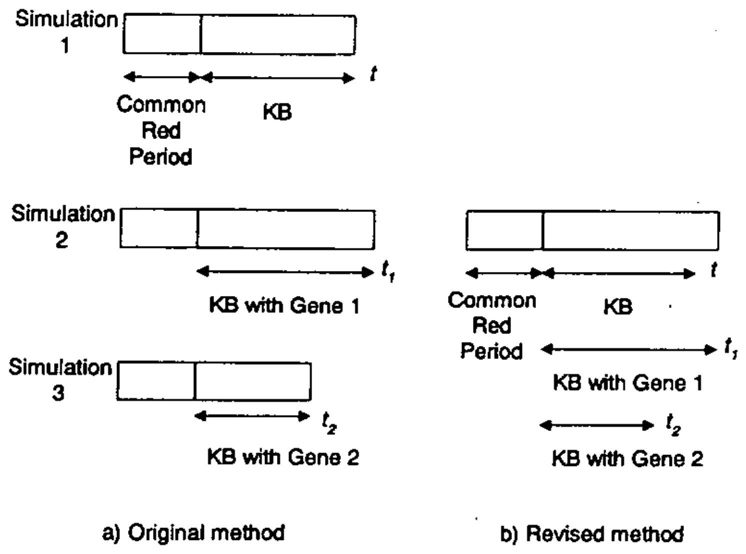

Figure 8.4 Space-time diagram of simulation speed up.

8.6 Results and Discussions of GA Learning

The GA learning is used to train the second level of the HFLC. The second level of the HFLC is a 2-input 1-ouput FLC and has 16 fuzzy labels in each of its parameter. Hence a complete control set will have 256 rules. The GA learning produces only 23 rules. The rules are listed in Table 8.2. The value shown in the table represents the fuzzy label shown in Figure 7.2.

The small number of rules generated indicates the first level of the HFLC alone is reasonably good. No significant modification on the effective green-time is needed in the second level of the HFLC.

f_j	$\sum_{i=1}^{4} f_i$	Δt
0	5	3
0	6	3
0	9	4
2	5	3
2	6	5
2	12	4
3	5	5
3	6	1
3	11	5
3	12	4
4	5	2
4	6	6
4	10	6
4	11	1
4	12	5
5	6	1
5	10	2
5	11	2
5	12	2
5	13	4
6	11	2
7	5	2

Table 8.2 Rule generated from GA learning process.

9 Dynamic Programming Coordination Control

9.1 Overview

A decentralized junction controller composed by a HFLC constructed previously and a coordination module is described in this chapter. The coordination module uses dynamic programming to determine optimal control policy for a junction. Technique to speed up the coordination control is also discussed.

9.2 Dynamic Programming Coordination Module

The purpose of the coordination module is to adjust the effective green time tg, obtained from the HFLC by a coordination parameter, t_c, so that a network optimal can be obtained. The network is optimal when the average delay and the number of stops are minimized. Dynamic programming (DP) is used to determine that coordination parameter.

With DP, a system is considered to progress through a number of consecutive stages. At each stage the system can be described by a set of parameters called state. At each stage, no matter what state the system is in, one or more decisions must be made in order to go to next stage. The possible decisions depend only upon the current stage and state. The past history of the system, i.e. how it got to the current stage and state, is of no importance. Moreover, the following condition must be satisfied: $t_g + t_c \geq 10\text{s}$

To speedup the calculation process, a limited number of stages is assumed. A phase of the signal cycle represents a single stage in the dynamic programming. The system transforms from one stage to another when the junction changes from one phase to another phase. At each stage, the HFLC determines an effective green time, t_g. Coordination between junctions is governed by coordination adjustment parameter, t_c. The sum of t_g and t_c is the final green time duration for that stage. The coordination adjustment parameter is determined by dynamic programming.

A state is a set of variables that describe the condition of the junction. The state variables, λ_s, consists of q_{si}, f_{si}, and t_s, where q_{si} is the queue length of traffic stream i, f_{si} is the flow-rates of traffic stream i at state s and t_s is the starting time of that state. Besides λ_s, there is one more parameter that is very important to determine transformation from one state to another, $\mathrm{E}[f_{si}(t_s : t_s + t_e)]$. It is the expected flow-rates of traffic stream i between time t_s and $t_s + t_e$, where $t_e = t_g + t_c$. This expected value is given by the junction connected to the traffic stream i and is the key coordination input. When a junction switches its phase, the HFLC will determine t_g based on q_{si} and f_{si}. The coordination control module then calculates a sequence of t_c for 4 successive phases such that the average delay and the number of stops per vehicle after 4 phases are minimized. In order to make such decision, the module has to know how many vehicles will enter the junction and when they will enter. This information $\mathrm{E}[f_{si}(t_s : t_s + t_e)]$ is exchanged between the adjacent junctions.

Assuming t_g is calculated 20s and the possible t_c is {-5s, 5s, 10s}, the coordination module will ask the adjacent junctions for the expected flow-rates after t = 20s+(-5s), 20s+5s and 20s+10s. It then evaluates the impact of having green time equals to t second. After 4 successive phases, the overall average delay and number of stops of all possible cases are calculated. The total green time producing minimal cost will be returned. Assuming the module has m possible actions at each phase. After n phases, the total number of possible actions will be m^n. If a tree diagram is used to represent all possible paths, there will be m^n nodes in the final level of the tree. Finding the optimal action requires m^n-1 comparisons. The computation intensity increases as the number of stages and states become larger. As a result, the decision set is limited in order to reduce the time complexity of determining the optimal:

$$t_c \in X_\lambda, X_\lambda = \{-10, -5, 0, 5, 10, 15, 20\}$$

Where X_λ is the possible decision set. A negative t_c means that the green time for the current phase has to be reduced by certain seconds. While a positive t_c means the green time has to be extended. When the resultant green time is less than 10 seconds, 10 seconds is assumed.

In order to get a balance on search space and computational intensity, m=7 and n=4 is chosen. The coordination module only considers successive 4 phases because a signal cycle in the model consists of 4 phases. Moreover, the highly fluctuating traffic environment makes traffic projection over long period. Decision set size equal to 7 is chosen since it produces a reasonable number of states. For example, decision set size of 8 produces 4096 different states while a size of 7 will gives 2401 states. There is however too little choice if the size of the decision set is equal to or smaller than 6.

9.2.1 State Merging

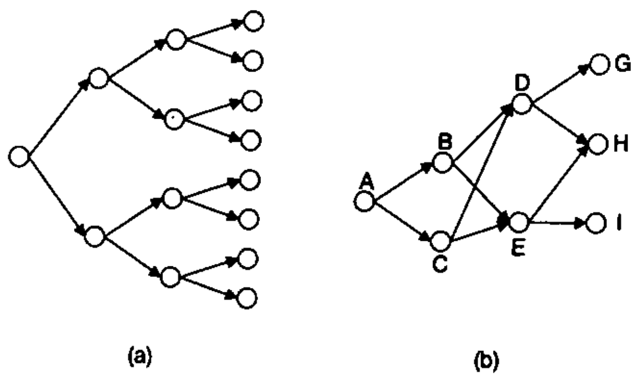

Figure 9.1 State diagram with and without state merging.

Figure 9.1a and b show the first 3 levels of the state tree when m=2 with and without state merging. Each node in the figure represents a single state of the junction at the time when a decision has to be made. Each arrow pointing out to the state shows how the state changes when different green times are applied. The green time is given by $t_g + t_c$ where t_c is the coordination parameter obtained by the coordination module.

States can be merged when they are equivalent. Two states are said to be equivalent if and only if:

- Their phase is the same, that is they are belonging to the same stage, and
- The start time of the phase t_s is equal, and
- Queue length and flow-rates of all traffic streams are similar.

Similar queue length means that the difference is less than 5 vehicles, which is equal to half the width of a fuzzy label as shown in Figure 5.6. Flow-rates are said to be similar when the difference is less than 0.25 veh/s and is also equal to half the width of a fuzzy label as shown in Figure 7.2. When 2 states merge, the larger value of queue length and flow-rates will be used. Since both the values of queue length and flow-rates are based on prediction, there is no way to know that the exact number of queues and flow-rates after applying green time = t_g + t_c seconds. As a result, approximate values are allowed.

As shown in the Figure 9.1a, there are 8 states in the final level and 14 ar rows in total. After state merging, the total number of arrows reduced into 10 and only 3 states left in the final level. The calculation time for DP is proportional to the number of links, each link has to be visited exactly once in the DP. When there is more than 1 link going into a node, comparison is also needed in order to determine the "cost" from the initial node to this node. The "cost" is the sum of average delay and number of stops per vehicle, and will be discussed later in this chapter. The prime purpose of DP is trying to minimize the "cost". If the "cost" from state A to D, AD, is required in Figure 11.1b, provided that the "cost" from A to B, AB, and from A to C, AC, are known. Then the "cost" AD, will be equal to the minimum of AB+BD and AC+CD. Similarly, AH = min{AD+DH, AE+EH} where AD = min{AB+BD, AC+CD} and AE = min{AB+BE, AC+CE}.

Assuming the time required for comparing and determining "cost" of an arrow is 1ms and 10ms respectively. Applying DP to Figure 9.1a, there are 14 arrows and 7 c omparisons at the final level to determine the optimal state. As a result, the time required for DP is 147ms. From Figure 9.1b, there are only 10 arrows, 3 nodes with multiple arrows, and 2 comparisons at the final level. The time required in 9.1b will be 105ms only. As a result, state merging reduces the time spent in finding the optimal path.

9.2.2 State Transformation

One of the important processes of dynamic programming is transformation. It represents how the state of the junction changes from one signal phase to another. With state transformation, a tree diagram can be constructed and hence DP determines the optimal route of the system. In this system, it is not possible to know exactly how many vehicles will come. As a result, assumption has to be made in order to construct the state diagram. Equation 9.1 shows transformation equation of the system. Assuming traffic stream *j* is given the right-of-way, the queue length of each traffic stream at next stage can be calculated by:

Equation 9.1 Stage transformation equation.

$$q_{(s+1)i} = \begin{cases} q_{si} + F_{si}T_{lost} + F_{si}(t_g + t_c) - d(t_g + t_c) & i = j \\ q_{si} + F_{si}T_{lost} + F_{si}(t_g + t_c) & i \neq j \end{cases}$$

Where $q_{(s+1)i}$ is the queue length of traffic stream *i* at stage *s+1*, $F_{si} = E[f_{si}(t_s : t_s + t_e)]$ and is given by adjacent junction connected to traffic stream *i*.

At each state, the average delay and total number of stops of all vehicles within the junction are calculated. They are the return of the dynamic programming coordination control of that stage. Path that traverses to current state with minimum average delay and total number of stops is also determined.

Equation 9.2 Cost function for DP.

$$\text{cost } z = D + S$$

Where *D* is the average delay per vehicle and *S* is the average number of stops per vehicle.

After 4 stages, the overall return, z, is calculated and the route with minimum cost is obtained. The overall return is calculated by equation 9.2. The corresponding values of t_c along the minimum route is the coordination control parameter of the junction. Figure 9.2 shows an example of state diagram of the system. For each stage, a minimum return and path to every state in the stage is calculated. For example, the minimum return and path to state *g* is from state *c*. At the final stage (stage 3), the overall minimum cost is obtained (state *j* in the figure) and path to the optimal state is returned (a, c, g, j). The flowchart of the coordination module is shown in Figure 9.3.

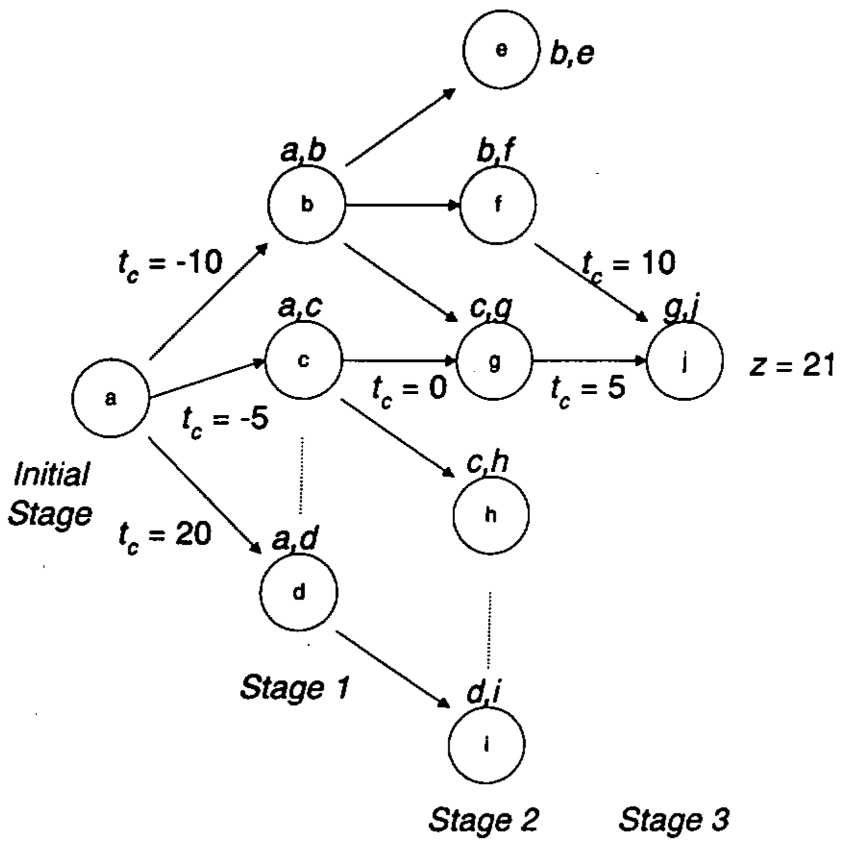

Figure 9.2 Sample state diagram of the dynamic programming coordination.

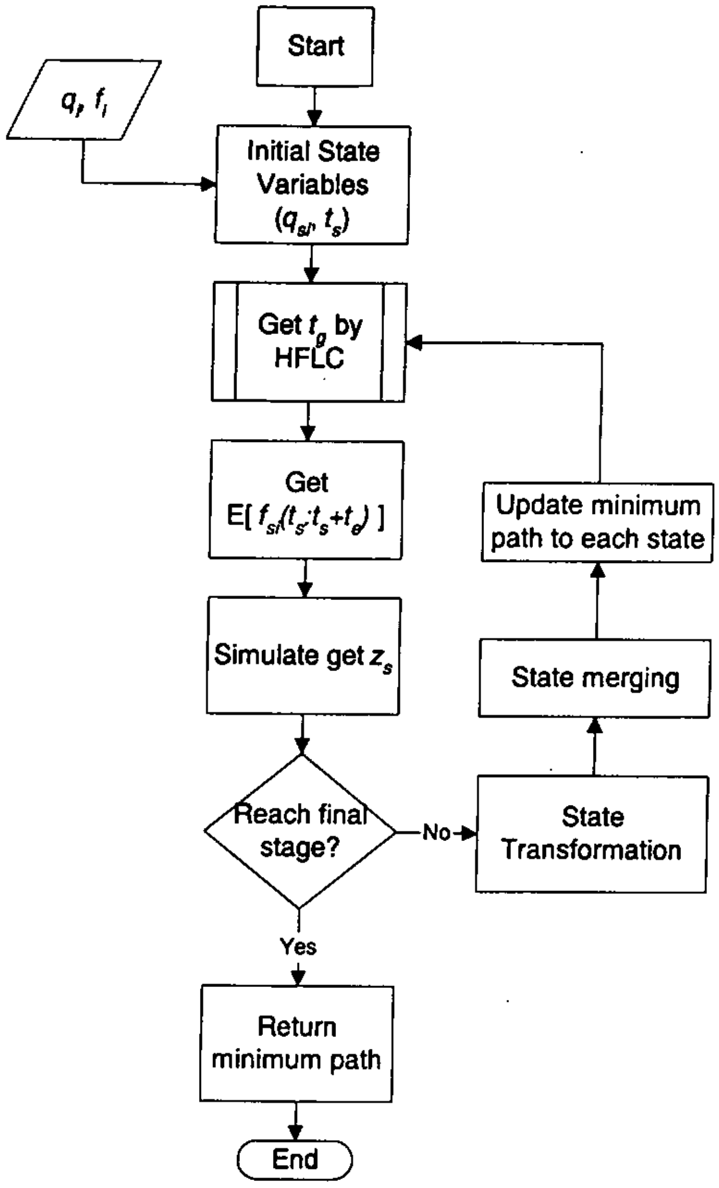

Figure 9.3 Flowchart of the Dynamic Programming Coordination Module.

9.3 Coordination Input - $\mathrm{E}[f_{si}(t_s : t_s + t_e)]$

As discussed earlier, $\mathrm{E}[f_{si}(t_s : t_s + t_e)]$ is the key coordination input of the DP and is given by the adjacent junctions. It is an expected flow-rate of traffic stream i between time t_s and $t_s + t_e$, where $t_e = t_g + t_c$. When determining the cost from stage s to $s+1$, the system has to make simulation in order to determine the average delay and number of stops per vehicle, and the state variable λ_{s+1} using equation 9.1.

During the period t_s to $t_s + t_e$, there are 4 possible cases:

1. The upstream junction is given a stop signal throughout the entire period.
2. The upstream junction is given a stop sign on or before time t_s and changes the signal within the period.
3. The upstream junction is given a right-of-way signal on or before time t_s and turn to stop signal within the period.
4. The upstream junction is given a right-of-way signal throughout the entire period.

The expected flow-rates of traffic stream in case 1 will be equal to 0, since there will be no vehicle entering the traffic stream. If the traffic stream is having the right-of-way, the DP will try to minimize t_c such that the green time will not be wasted. The expected flow-rates of all other cases will be equal to the mean discharge rate of that stream.

For example, suppose the downstream junction ask for the expected flow-rates where t_s=20s and t_e=15s, the upstream junction will determine which case it is in according to the last cycle length. Then the expected value, $\mathrm{E}[f_{si}(t_s : t_s + t_e)]$, will be returned to the downstream junction. The downstream will then make use of this value to calculate the corresponding cost.

9.4 Objective Function

The overall network cost, z, is defined as weighted sum of delay and stops of the traffic network. Assuming there are n junctions in the network. The goal is to minimize z

such that:

Equation 9.3 Objective function for DP.

$$\text{Min} \quad z = \sum_{k=1}^{n} (D_k W_D + S_k W_S)$$

Where D_k is the average delay (in seconds) per vehicle in junction k, W_D is the overall delay weighting, S_k is the average number of stops per vehicle in junction k, W_S is the overall stops weighting.

Equation 9.4 Average delay.

$$D_k = \frac{\sum_{i=1}^{4} d_{ik} f_{ik}}{\sum_{i=1}^{4} f_{ik}}$$

Equation 9.5 Average number of stops.

$$S_k = \frac{\sum_{i=1}^{4} s_{ik} f_{ik}}{\sum_{i=1}^{4} f_{ik}}$$

Where d_{ik} and s_{ik} are the average delay per vehicle and average number of stops per vehicle at traffic stream i of junction k respectively, f_{ik} is the flow-rate of traffic stream i in junction k.

In determining the cost of single junction when using DP, equation 9.2 is used instead.

9.5 Enhancement in Dynamic Programming

According to chapter 9.2.1, the speed of the DP is greatly affected by the number of states in the state tree. State merging is used to speed up the DP. Suppose a stage has a total of n states, each state has to compare to all others in order to merge 2 states together. As a result, $(n-1)^2$ comparisons have to made at each stage. To speedup the process, hash table storing t_s of each state is used. When a state is created, the program will take a look in the hash table to see whether there is other state having the same t_s. Comparison with all other state variables is then made. Suppose there is at most m states having same t_s, the number of comparisons required when using hash table is $m(n-1)$ where $m<n$. As a result, state merging is speed up.

To speed up the simulation time of the determination of the cost of each link, technique discussed in chapter 8.5 is also used. Since each state can have at most 7 different t_c to choose as described in chapter 9.2, 7 different simulations have to be performed to determine the cost of each route. However, the simulations can be combined together so that only 1 is required. Detail description of the process can be found in chapter 8.5.

9.6 Conclusions

A junction coordination module is constructed. Dynamic programming is used to optimize the module so that the output of the system will produce minimum traffic delay. As an optimization process is required in every phase of a signal cycle, techniques to speed up the process are introduced. The number of possible actions in each phase is limited. The size of the state diagram can be reduced and hence the time needed for the optimization process is also reduced.

A complete decentralized junction controller is now constructed. It includes a HFLC for controlling local traffic and a dynamic programming optimization for junction coordination. Simulation on the controller is performed and result is discussed in chapter 12.

10 Simple HFLC

10.1 Overview

Computer simulation is used to test the performance of the simple HFLC constructed in chapter 6. The controller is subject to constant and time varying traffic conditions. Simulation will execute 5 times for each traffic condition. They have different arrival characteristics because the vehicles arrive randomly. The mean of the average delay attained from the 5 simulations is calculated and used to evaluate the performance of the system. The HFLC is proved to be better than conventional fixed-time control in reducing average delay per vehicle under both constant and time-varying traffic conditions.

10.2 Simulation Method

Vehicle arrival pattern is generated first, their pattern randomly distributed with its mean equal to the flow-rate. Moreover, each vehicle can either turn left, right, or drive straight when they pass the junction. The turning percentages as well as the flow-rates are pre-defined but can be changed during the simulation.

In every 10 seconds, average delay for vehicles in the network is calculated. It is recorded and plotted against time. At the end of each simulation, the overall average delay is calculated and used as the performance indicator.

The simulation generates a new set of traffic profile each time. Since the traffic profile is generated randomly with the mean equal to the pre-defined flow-rate, every run will be slightly different and the mean of the average delay in 5 runs is calculated.

10.3 Constant Traffic Condition

Under constant traffic condition testing, the system is required to simulate 10 hours of traffic. The extensive simulation time is to test the stability of the controller over a long period of time.

The HFLC is compared against fixed-time controller with optimum cycle time calculated. Both HFLC and fixed-time controller are tested using different combination of flow-rates from light (50% to maximum junction capacity) to heavy traffic (80% to maximum junction capacity). 4 different cases under the same junction capacity are simulated: with a dominant traffic stream, 2 dominant traffic streams, 1 traffic stream with relatively low flow-rate, and all the traffic streams' flow-rate are similar. Results of the simulation runs are listed in Table 10.1.

The case with 1 single dominant traffic stream represents the directional flow of major road during morning or evening rush hour. For example, during the morning rush hour, most people try to get on a major road and go to office (e.g. city center). As a result, majority of vehicles go to the same direction and hence a single dominant traffic stream occurs in the junction. When 2 of these directional major roads come across or at the intersection of a major road and a small road, it will produce a junction with 2 dominant traffic streams. A junction with a relatively low flow-rate traffic stream represents the case when there is a minor branch going to suburban area. The traffic connecting to those areas is light. The case for similar traffic stream flow-rate simulates junction at city center that every traffic stream is congested.

Table 10.1 Average delay per vehicle in simple HFLC.

Junction [illegible] capacity [illegible]	Cas[illegible]	Fixed time [illegible] controller [illegible] (sec/veh)	HFLC [illegible] (sec/veh)	Reduction [illegible] (%)
40%	A	33.2428	32.1723	3.2202
	B	33.9922	33.9564	0.1053
	C	33.2394	33.2229	0.0496
	D	33.5637	33.2492	0.9558
50%	A	36.5247	36.3425	0.4988
	B	38.0612	35.7000	6.2037
	C	39.1067	38.4383	1.7092
	D	42.6127	41.2561	3.1836
60%	A	50.6566	49.3203	2.638
	B	48.0681	43.2539	10.0154
	C	48.2599	44.3631	8.0746
	D	48.7297	47.8048	1.8980
70%	A	59.8571	59.1175	1.2356
	A'	*79.8856*	*59.1175*	*25.9973*
	B	59.3651	53.3305	10.1652
	B'	*72.3217*	*53.3305*	*26.2593*
	C	63.5353	59.1192	6.9506
	D	66.0414	59.2484	10.2860
80%	A	99.2988	95.9826	3.3396
	B	97.3981	92.9547	4.5621
	C	96.3131	90.9435	5.5752
	D	112.7153	106.8087	5.2403

* Case A:	Junction with single dominate traffic stream.
Case A':	Junction with single dominant traffic stream with 3 seconds less in the green time of the non-dominant traffic streams.
Case B:	With 2 dominate traffic streams.
Case B':	2 dominant traffic streams with 3 seconds less in the green time of the non-dominant traffic streams.
Case C:	With 1 relative low flow-rate traffic stream.
Case D:	Flow-rates of all traffic streams are similar.

When junction capacity is smaller than 40%, the calculated effective green time of each phase is always less than 10s, which violates the assumption in chapter 5.4. Hence, 10s of the effective green time is always given without invoking the HFLC. As a result, simulation on less than 40% of junction capacity is not shown.

From Table 10.1, when the traffic is light, there is not much difference on the small changes in the effective green time. As a result, reduction in average delay is not significant. On the other hand, the result also proved that the fixed-time control is very near to optimum under light traffic. The result is very different when traffic is medium (around 60-70% of junction capacity). As shown in cases A and A', the average delay changes significantly in only few seconds difference on the effective green time. As a result, the HFLC performs best over this region. However, in the case of heavy traffic, since there is not much room for further reduction, the reduction on the average delay drops again.

Effect in changing the effective green time for the fixed-time control is demonstrated using the 70% of junction capacity (case A' and B'). Effective green time for all the non-dominant traffic streams is reduced by 3 seconds. Delays are increased significantly in the fixed-time control. Figure 10.1 shows the average delay against time in the case A' at 70% of junction capacity.

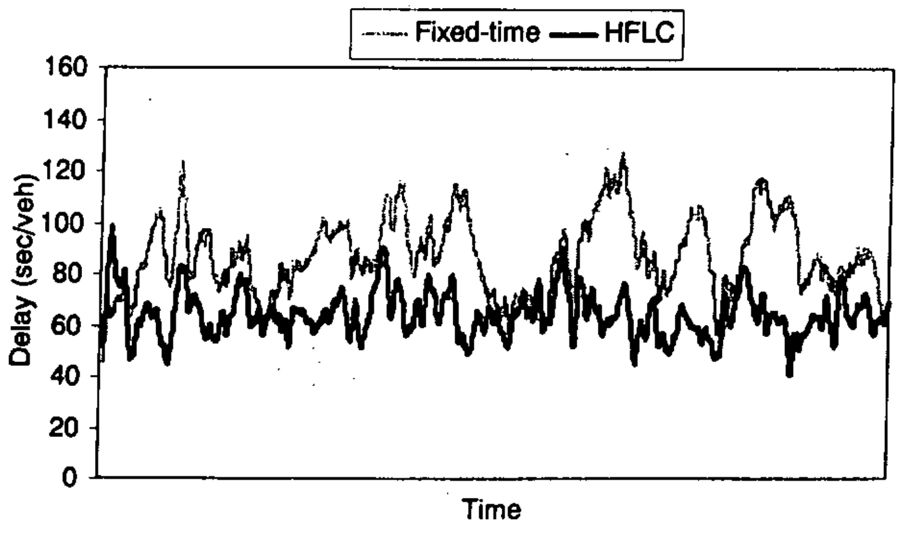

Figure 10.1 Average delay variation of case A under 70% of junction capacity.

At the beginning of the simulation, the average delay for the fixed-time control increases gradually as queue builds up slowly. The flow-rate prediction of the HFLC equals to the weighted combination of the pervious and present detected flow-rate given in equation 10.1. At the very beginning of the simulation, the pervious flow-rate is equal to 0. This leads the HFLC to believe that the flow-rate is very small and hence makes a wrong decision by giving a very short effective green time. As a result, the average delay for the HFLC is relatively higher than the fixed-time control at the beginning of the simulation. After a few cycles, the flow-rate determination error reduced and hence the HFLC can make better decision and the average delay is reduced.

Equation 10.1 Flow-rate prediction equation.

$$f' = 0.7 \times f + 0.3 \times f_{\text{detect}}$$

Where f' is the predicted flow-rate, f is the pervious flow-rate, and f_{detect} is the present detected flow-rate.

10.4 Time-varying Traffic Condition

In order to demonstrate the capability on handling time-varying traffic demand, the controller is subject to a demand pattern similar to the traffic within one full day. The pattern is divided into 4 time periods, namely "night time (NT)", "morning peak (MP)", "day time (DT)" and "evening peak (EP)". The traffic data is referencing from traffic survey held by Transport Department in Hong Kong [4].

Table 10.2 Time sections for fixed time controller.

Night time (NT)	10:30 p.m. – 08:00 a.m.
Morning peak (MP)	08:00 a.m. – 10:30 a.m.
Day time (DT)	10:30 a.m. – 04:30 p.m.
Evening peak (EP)	04:30 p.m. – 10:30 p.m.

Similar to constant traffic condition testes, the controllers are subject to 4 different cases:

- (Case A) Single dominant traffic stream
- (Case B) 2 dominant traffic streams
- (Case C) 1 traffic stream with relatively low flow-rate
- (Case D) Flow-rates of all traffic streams are similar

In each case, 5 tests are performed and the mean of average delay per vehicle is recorded. Table 10.3 shows the average delays under different time zones.

Table 10.3 Average delay of vehicle in second under different time zones.

Case	Time zone	Fixed-time controller (sec/veh)	HFLC (sec/veh)	Reduction (%)
A	NT	36.5032	28.8730	20.9
	MP	95.8950	74.0067	22.83
	DT	60.2427	56.7650	5.77
	EP	96.6423	72.4106	25.07
	Overall	77.0096	62.1220	19.33
B	NT	38.3513	26.1293	31.87
	MP	95.9881	87.1162	9.24
	DT	72.3505	64.1591	11.32
	EP	91.1812	75.7660	16.91
	Overall	79.7935	68.6505	13.96
C	NT	43.1101	25.8034	40.15
	MP	102.395	97.0965	5.17
	DT	69.6700	68.0841	2.28
	EP	98.7502	84.7532	14.17
	Overall	83.1499	74.8986	9.92
D	NT	37.0213	26.1162	29.46
	MP	92.1696	81.4981	11.58
	DT	69.0380	60.7693	11.98
	EP	89.4060	73.9385	17.3
	Overall	78.6786	64.4514	18.08

As shown in Table 10.3, the HFLC reduces the overall average delay for more than 9%. The reduction is more significant when traffic is light, more than 20% reduction is achieved. It has been shown in the previous section that the HFLC performs well when traffic is constant. As a result, the handling of time-varying condition is the prime concern in this section. Figure 10.1 shows average delay against time in one of the simulations.

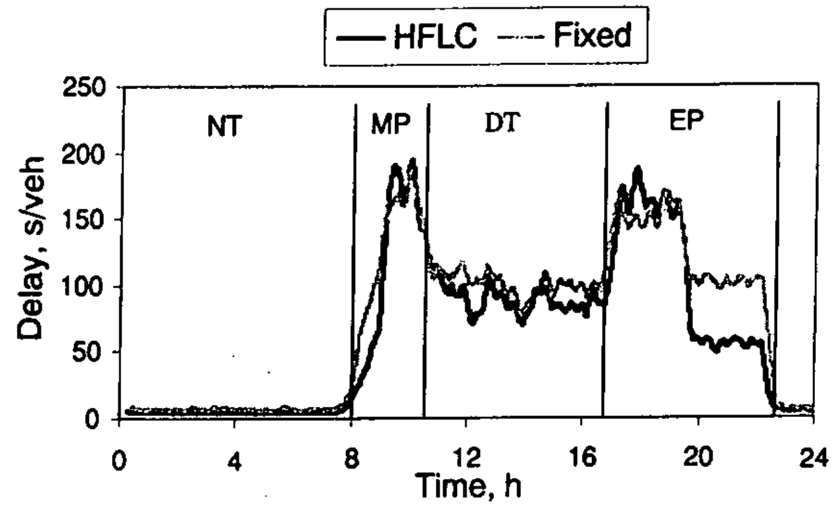

Figure 10.2 Average delay per vehicle of HFLC and fixed-time control for junction with single dominant traffic stream.

From Figure 10.2, it can be noted that under the non-peak time zones (NT and DT), average delay per vehicle using the HFLC is smaller than that using the fixed-time control. It is because under the non-peak zones, the flow-rates for all traffic streams are quite steady, which is similar to the results under constant traffic condition. Table 10.1 already shows the HFLC is better than the fixed-time control when traffic is constant.

Under the peak periods (MP and EP), the HFLC reduces average delay from 5 to 22%. However, figure 10.2 shows there exists some situations in which the average delay attained from the HFLC is greater than that from the fixed-time control, especially when the flow-rates reaches its maximum. It is because the HFLC does not handle traffic well at those time-varying regions. From the architecture of the HFLC, the first level is derived from equation (4.5), which does not consider the time-varying property of the traffic and the coordination among different traffic streams. All are embedded into the second level of the HFLC, where the rule set for this level is derived mainly from trial-and-error process. From this time-varying experiment, it can be concluded that the rule set for the second level of the HFLC is not optimal and hence a better way to obtain the rule set is needed. As a result, genetic algorithm (GA) will be used to obtain the rule set automatically without any human interaction.

In the second half of EP, average delays of both the fixed-time and HFLC drop significantly. It is because the junction capacity during the second half of EP is less than that in the first half. The fixed-time controller, without previous knowledge in the reduction in flow-rate, introduces more delay to the vehicles than the HFLC does. Although the flow -rates pattern is very similar day-by-day, if there is a sudden change in the pattern, the fixed-time control will fail to produce suitable control action unless re-construction of the signal plan is performed. The HFLC, with the help of vehicle detectors, can adapt to these changes comfortably without re-building the rules.

10.5 Discussions

The results show that the HFLC achieves delay reduction in general. During constant traffic condition, the HFLC reduces average delay per vehicle by about 3 to 4% when compared with a fixed-time controller. In time-varying traffic condition, the HFLC can reduce overall average delay up to 18%. However, in the transient period where the rate of change of flow-rate is relatively large, the performance of the fixed-time controller is better. This indicates that the rule set for the second level of HFLC, which is currently derived by human trial-and-error technique, is not optimal. A more systematic way in deriving rules for the second level is needed. Genetic algorithm (GA), a heuristic multidimensional search algorithm, will be used to derive an optimal set of rules for the second level of HFLC.

The time reduction is not very significant when it is converted back into travel time, only few seconds are reduced and the driver will not even notice it. However, this is just the case for 1 single junction. For multiple junctions, the total time reduction can be large by summing up every individual junction delay. Moreover, the cycle time for fixed-time controller is derived from the exact flow-rate of the junction. This is not true in the real practice because engineer must allow some buffer for small growth of the traffic demand. Otherwise, the cycle time may need to revise frequently. As a result, the reduction by the HFLC can be much greater.

Unlike the fixed-time controller, the HFLC does not require the knowledge of the flow-rate pattern to attain suitable effective green time, which makes the HFLC capable of handling unexpected changes of traffic demands.

To summarize, simulation results encourage further studies on HFLC. In the next chapter, the HFLC with GA learning will be tested.

11 HFLC with GA Learning

11.1 Overview

In chapter 7, GA learning has been incorporated in the HFLC. The first level of the HFLC is a 2-input 1-output FLC with the rules derived from equation 5.3. The second level is also a 2-input 1-output FLC but the rules are derived from automatic GA learning. The GA learning process produces only 23 rules. Simulation results are given here to demonstrate the performance of the HFLC with and without the GA learning process under various traffic conditions.

11.2 Constant Traffic Conditions

The HFLC is first trained and the performance of HFLC with and without GA training will be compared, as well as against that of a fixed-time controller, through simulation. Similar to the simulations mentioned in the last chapter, 20 different tests are performed and the mean of average delay among them is recorded. Table 11.1 shows result of the tests. It is clear that the HFLC reduces the average delay with and without GA training. Reduction with GA training is particularly notable.

As shown in Table 11.1, the HFLC undoubtedly outperforms the fixed-time controller in term of traffic delay reduction. On average, the HFLC before GA training reduce 0 to 8% traffic delays. It is because the HFLC before training acts like a real-time controller using the Webster optimal cycle equation [42]. As a result, reduction is not significant. The controller only takes advantage on the slight flow-rate fluctuation. By comparing Table 11.1 and Table 10.1, even the HFLC without GA training performs better than the one in last chapter. The increase of the number of fuzzy labels and hence a more accurate model certainly helps.

After the GA training, it is found that the reduction percentage is increased significantly. The increase is notable under heavy traffic, and more than 10% reduction is achieved. The average delay reduction after the training increases gradually as the capacity increase. It is because there is no much space to optimize when the flow-rate is light. The effect is more significant under heavy traffic condition. The improvement of result shows that GA improves the rules under the HFLC in constant traffic conditions.

Table 11.1 Average delay (s/veh) by the fixed-time controller and the HFLC with and without GA learning under constant flow-rates.

Junction capacity	Fixed time (s/veh)	HFLC (s/veh) (before training)	HFLC (s/veh) (after GA training)	Reduction in seconds before and after training	Reduction (%)	
					Before training	After training
40%	29.31	29.03	28.91	0.12	0.955305	1.364722
45%	31.8	30.5	30	0.5	4.08805	5.660377
50%	35.83	34.61	33.42	1.19	3.404968	6.726207
55%	39.68	38.1	36.86	1.24	3.981855	7.106855
60%	44.56	44.15	42.39	1.76	0.920108	4.869838
65%	53.18	50.12	47.08	3.04	5.754043	11.47048
70%	64.01	63.6	59.3	4.3	0.640525	7.358225
75%	73.92	70.07	65.94	4.13	5.208333	10.79545
80%	92.37	90.37	76.94	13.43	2.165205	16.70456
85%	129	118.88	103.59	15.29	7.844961	19.69767
90%	203.22	201.51	175.22	26.29	0.841453	13.77817

11.3 Step Changes in Traffic Condition

From the results in the last chapter, the previous design of the HFLC performs rather disappointingly under transient conditions, where the rate of change of flow-rate is high. For the testing purpose, the controller is subject to a step change of traffic demand. A step increase of flow-rate followed shortly by a step decrease is imposed to one of the traffic streams, Φ_3. Initially, the total flow-rate of the junction is set to 0.7 veh/s. The increase in flow-rate makes the junction over-saturated (total flow-rates = 1.1 veh/s) for 3 minutes.

As shown in the Figure 11.1, the HFLC enables faster recovery from sudden disturbance. Hence the effects of the disturbance are alleviated. Moreover, the peak average delay in the HFLC is also lower than that of the fixed-time control.

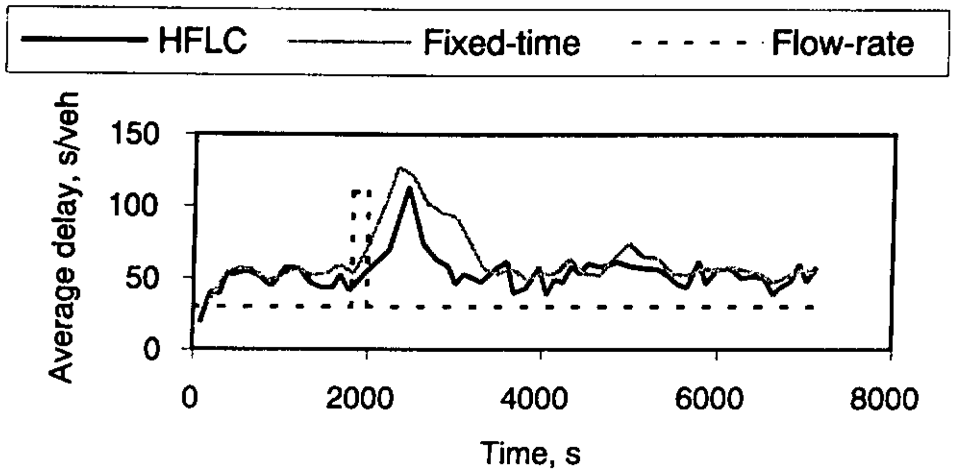

Figure 11.1 Overall average delay of HFLC and fixed-time under step change flow-rates.

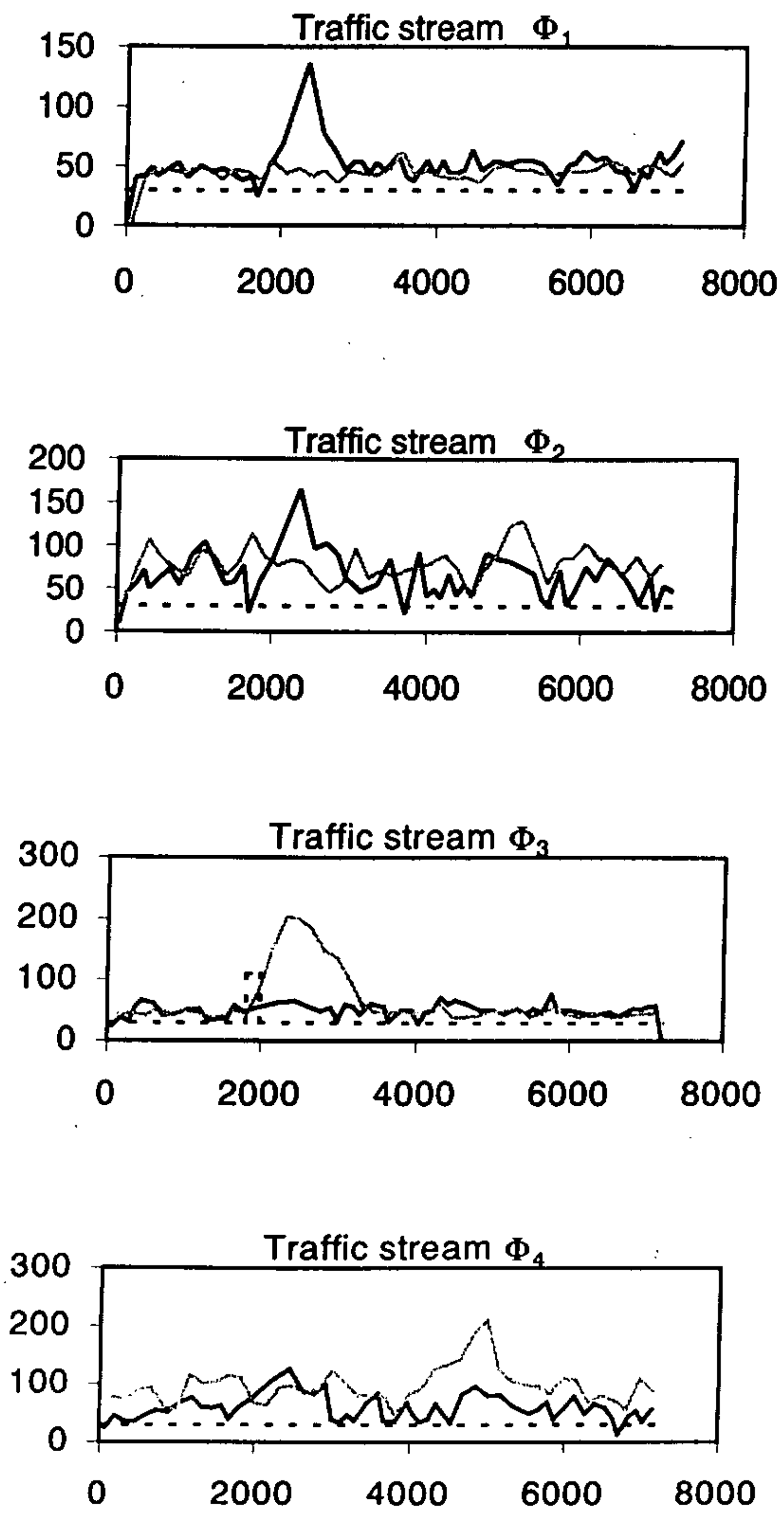

Figure 11.2 Average delay at all traffic streams.

From Figure 11.2, it can be observed that penalizing or spreading the delay on other traffic streams having smaller traffic demand achieves reduction of overall average delay.

11.4 Time-varying Traffic Conditions

Next, the controller is tested under time varying traffic conditions where the traffic demand is similar to the real situation. Similar to chapter 10.4, flow-rate profile similar to one-day's traffic pattern [4] is used. Figure 11.3 shows a simplify view of the data used in simulation. The profile is divided into 4 periods, namely "NT", "MP", "DT" and "EP". A cycle time plan for each section is calculated for the fixed-time controller who switches the time plans according to the time of day.

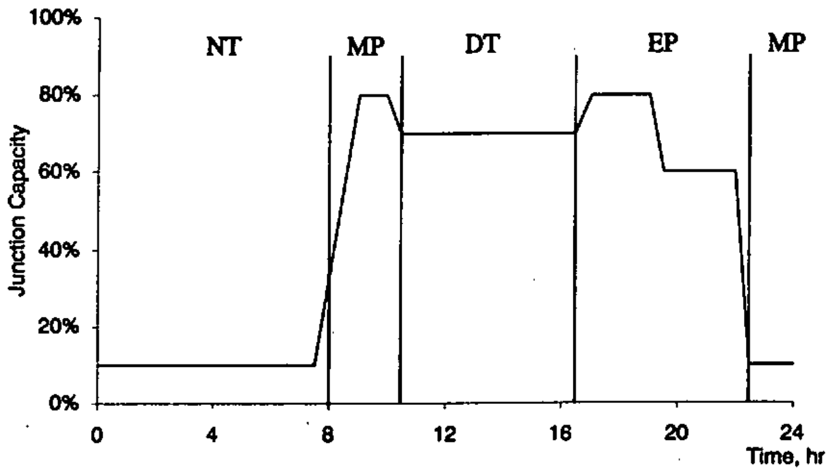

Figure 11.3 Flow-rates pattern for continuous flow-rates changing test.

Similar to the last chapter, the controllers are subject to 4 different cases:

- (Case A) Single dominant traffic stream
- (Case B) 2 dominant traffic streams
- (Case C) 1 traffic stream with relatively low flow-rate

(Case D) Flow-rates of all traffic streams are similar

In each case, 5 tests are performed and the mean of average delay per vehicle is recorded. The average delays in different time zones under different cases are given in Table 11.2 and one of the delay variations is shown in Figure 11.4.

Table 11.2 Average delay in second under different time zone.

Case	Time zone	HFLC (sec/veh) (before training)	HFLG (sec/veh) (after GA training)	Reduction (%)
A	NT	28.8730	26.2346	9.14
	MP	74.0067	69.0379	6.71
	DT	56.7650	55.7810	1.73
	EP	72.4106	63.3512	12.51
	Overall	62.1220	59.9372	3.51
B	NT	26.1293	25.7151	1.59
	MP	87.1162	74.1379	14.90
	DT	64.1591	56.9819	11.19
	EP	75.7660	67.3792	11.07
	Overall	68.6505	60.3658	12.07
C	NT	25.8034	23.6124	8.49
	MP	97.0965	73.1477	24.66
	DT	68.0841	56.0919	17.61
	EP	84.7532	66.6180	21.40
	Overall	74.8986	59.5763	20.46
D	NT	26.1162	25.5603	2.13
	MP	81.4981	73.7393	9.52
	DT	60.7693	55.6920	8.36
	EP	73.9385	68.6667	7.13
	Overall	64.4514	60.0992	6.75

Table 11.2 compares the performance of the HFLC with and without GA optimization. From the results, GA optimization shows further reduction on average delay, especially under heavy traffic. In general, the increase in the number of fuzzy label enhances the performance of the system over the constant flow-rate region ("NT" and "DT"). Moreover, during the peak periods ("MP" and "EP"), the transient effect due to the changes in flow-rate is reduced by the rules generated from the GA. Besides, it is clear that the percentage reduction on the peak periods are greater than the non-peak periods. This phenomenon is reflected by the fact that the improvement from GA training is greater than that from the increase of the number of fuzzy labels.

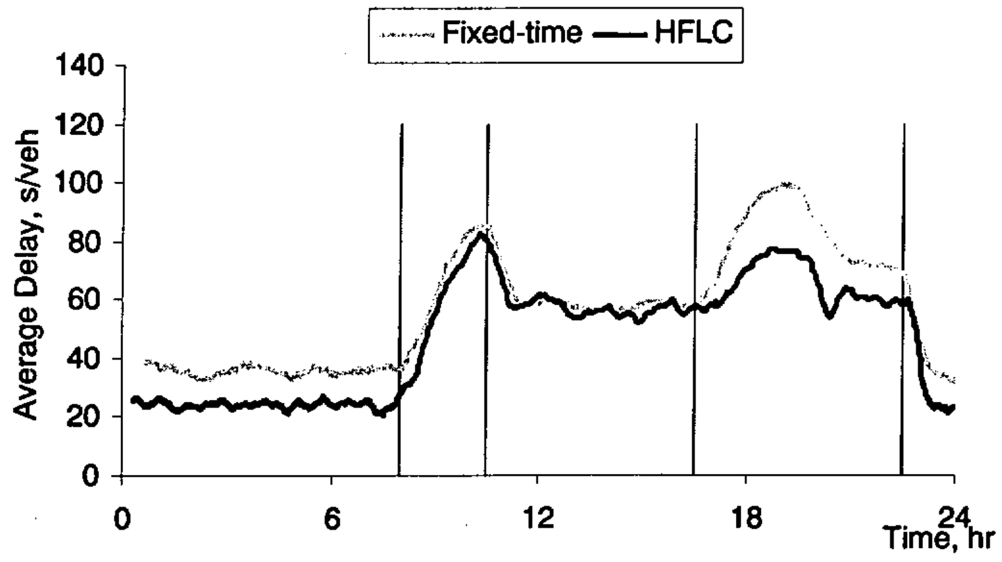

Figure 11.4 Average delay of HFLC with GA training and fixed-time under time-varying traffic environment.

As shown in Figure 11.4, the recovery of the HFLC from peak congestion is faster (similar to results obtained from step changes traffic environment). The average delay within the 4 time zones has been reduced, particularly around the peak-hours. There are 2 peak sections in which the flow-rates are high, "MP" and "EP". It should be noted that the maximum flow-rates at these 2 peaks are the same but the average delay produced by the HFLC in the "MP" is greater than that in the "EP". It is because the rate of change of flow-rate in the "MP" (from 0.1 veh/s to 0.8 veh/s) is much higher than that in the "EP" (from 0.7 veh/s to 0.8 veh/s). Unlike what was shown in Figure 10.2, the maximum average delay of the HFLC during peak hour is no longer higher than that from the fixed-time.

11.5 Discussions

To eliminate the human trial-and-error rule-definition process, GA is used to automatically generate the fuzzy rules. Simulation results show that with the help of GA, average delay per vehicle is significantly reduced, especially over the transient region. Moreover, the GA is capable of producing a minimum set of rules necessary to the system. All unnecessary rules are removed by the natural selection function of GA.

GA is an evolution of searching algorithm. There is no way to tell when exactly the training is complete. As a result, the GA training is only an off-line process. Since the output of the GA training is the fuzzy control rules, a controller can be first trained off-line and then deployed over other junctions with similar structure.

11.6 Conclusions

A HFLC for isolated junction has been successfully designed, built and tested. The HFLC consists of 2-level with a 2-input 1-output fuzzy logic controller on each level. The rule for the first level is derived purely from a mathematical model while the second level is obtained by off-line GA learning. No human interaction is needed for the rule definition process.

The HFLC is capable of handling any traffic conditions without prior knowledge or prediction on traffic pattern. Simulations show that the HFLC reduce traffic delay significantly over traditional fixed-time controller. Moreover, with the help of GA training, optimal control is achieved in both static and time-varying traffic conditions.

12 Decentralized Control

12.1 Overview

A decentralized area traffic control has been developed. It consists of a HFLC for local traffic control and a dynamic programming junction coordination control module. The junction coordination module determines a set of control actions that produce minimal average delay over a period of time. Simulation is carried out to test the decentralized area traffic controller.

12.2 Simulation Model

A simple network consist of 2 junctions is constructed as shown in Figure 12.1. Vehicle detectors are installed in front of and in vicinity of the junctions. The vehicle detectors are shared between 2 junctions in the common link. For example, the detector Y acts as junction-approach detector of junction A and stop-line detector of junction B. This setup can reduce the total number of vehicle detector used and hence reduce the cost of system installation. Priority traffic is the right-to-left direction, i.e. from junction B to junction A. In other words, delay and number of stop of vehicles traveling from junction B to A have higher weights in the cost function. The common link can hold up to 50 vehicles in each direction. The turning percentage of all cases is set to 10% (both left and right turn). Vehicle arrives randomly in all traffic streams connected outside with means equal to the flow-rate. In the common link, vehicles are moving in platoon and no dispersion is assumed. As a result, the optimal offset of junctions having common cycle length is the traveling time of vehicle in the common link.

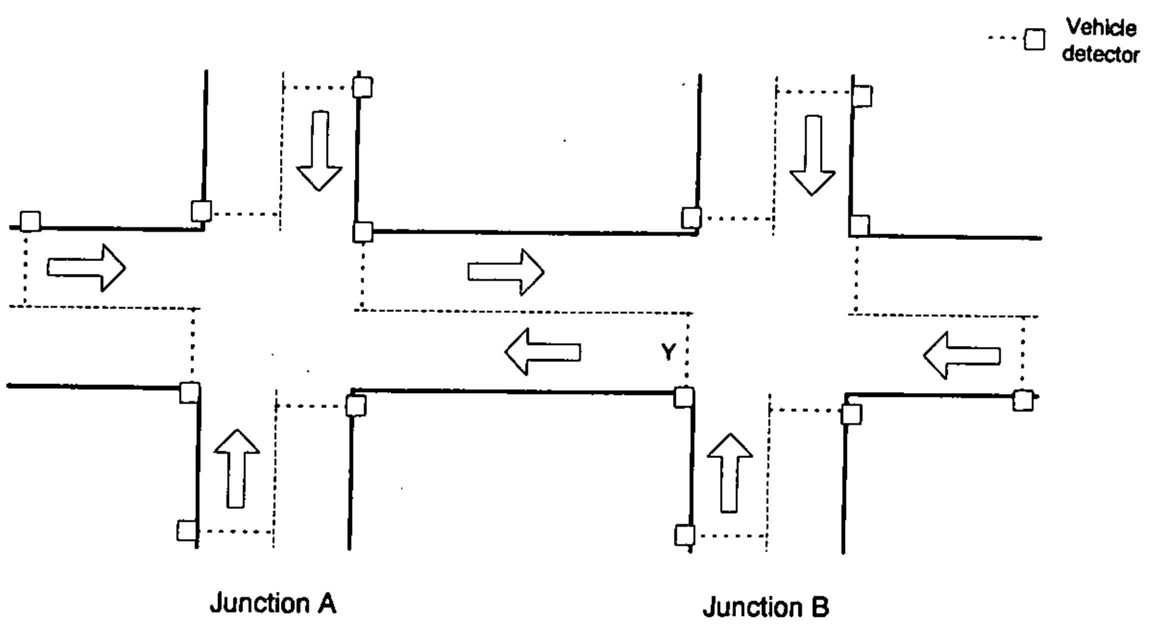

Figure 12.1 Junctions Setup.

12.3 Constant Traffic Condition

The network is first tested with constant traffic. The main objective of this simulation is to test the dynamic programming optimization. The result obtained depends on the accuracy of the state diagram. Obviously, the exact flow-rates of the junction in the future is not known, their expected value, $E[f_{si}(t_s : t_s + t_e)]$, is used to construct the state diagram by using equation 9.1.

Similar to the performance evaluation for single junction controller, a fixed-time control is constructed as reference for comparison. The initial offset for the fixed-time control is set to the average cruising time from junction B to junction A. This value is kept unchanged throughout the simulation. Moreover, the cycle time of junction depends on local junction flow-rates only. As a result, if the flow-rates of the 2 junctions are different, the cycle time will also be different. As time goes on, the offset difference becomes larger. Until the difference is divisible by the cycle length, they will have the optimal offset again. If both cycle times are equal, common green time with optimal offset is achieved. Vehicles coming from junction B do not need to stop when entering junction A. Hence the fixed-time control will produce optimal results when the flow-rates of 2 junctions are equal.

Different combinations of flow-rates in junction A and B are tested. Simulation results are shown in Tables 12.1 to 12.3. The horizontal axis of tables is the sum of flow-rates of junction B and the vertical axis is the sum of flow-rates of junction A. All the flow-rates are in percentage relative to saturation flow. The cost *z*, is calculated by equation 12.1, and the objective is of course to minimize *z*.

Equation 12.1

$$z = \left(\frac{(D_A + S_A) + (D_B + S_B)}{2} + d_P \right) / 2$$

Where D_A and D_B are the average delay in second per vehicle of junction A and B respectively. S_A and S_B are the average number of stop per vehicle of junction A and B respectively. d_p is the average delay of vehicle over the priority route, traffic going from junction B to junction A.

Equation 12.2

$$D_A = \frac{\sum_{i=1}^{4} d_{iA} f_{iA}}{\sum_{i=1}^{4} f_{iA}}$$

Equation 12.3

$$S_A = \frac{\sum_{i=1}^{4} s_{iA} f_{iA}}{\sum_{i=1}^{4} f_{iA}}$$

Where d_{iA} and s_{iA} are the average delay per vehicle and average number of stop per vehicle in traffic stream i at junction A. f_{iA} is the flow-rate of traffic stream i at junction A.

The cost function of equation 12.1 and 11.2 is different. Equation 11.2 represents local optimal from junction point of view, while equation 12.1 gives a network optimal. From the local point of view, there is no way to determine the value of d_p. As a result, the local controller will try to obtain an optimal on all traffic streams. However, a centralized controller can determine this value since it considers the whole network at a time.

Table 12.1 Cost z of the fixed-time control.

Junction A	Junction B										
	0.45	**0.50**	**0.55**	**0.60**	**0.65**	**0.70**	**0.75**	**0.80**	**0.85**	**0.90**	**0.95**
0.40	50.9	52.4	51.8	56.8	63.8	77.4	90.9	127.6	342.4	437.4	1073
0.45	34.1	54.3	54.4	60.4	56.5	68.6	88	109.8	173.7	210.3	546.7
0.50	53.5	37	58	63.1	55.8	63.8	82.6	100.3	213.7	338	421.1
0.55	57.8	58.9	39.2	69.4	111	100.4	117.4	429.5	753.7	992.1	1272
0.60	56.4	60	62.7	42.4	80.3	97.2	271.1	128.1	497.3	754	1046
0.65	58.4	62.9	64.7	69.1	49.8	81.2	210.5	131.9	179	619.3	772.3
0.70	60.4	64	69.2	73.6	77.5	50.9	111.4	306.3	627.6	495.2	732.1
0.75	76.4	79.6	74.9	81.7	96.3	116.9	59.8	147	161.8	403.6	729.9
0.80	68.5	99.9	84.1	118.7	210.1	189.7	374.7	72.1	163	206.9	153.5
0.85	92.3	103.5	87.6	135.5	183.4	242.1	444.5	566.6	79.9	495.4	698.3
0.90	153.9	153.9	168.8	185.2	275.3	387.8	468.1	693.9	373.7	196.6	745.6

Table 12.2 Cost z of the DP decentralized control.

Junction A	Junction B										
	0.45	**0.50**	**0.55**	**0.60**	**0.65**	**0.70**	**0.75**	**0.80**	**0.85**	**0.90**	**0.95**
0.40	35.28	37.64	38.96	44.84	55.52	73.18	78.12	86.22	102.6	119.2	121.8
0.45	40.4	41.74	46.72	52.16	54.7	75.32	83.24	103.5	126.3	165.9	243.6
0.50	44.32	48.32	46.98	57.46	60.6	70.56	83.56	103.4	125.5	191.7	265.6
0.55	53.12	53.34	59.48	53.64	58.3	64.26	74.04	85.2	111.6	145.6	139.3
0.60	58	62.7	66.44	70.2	67.2	79.72	94.34	81.58	96.48	149.7	173.2
0.65	64.9	65.34	73.54	80.76	78.85	76.86	88.54	90.38	104.2	171	164.8
0.70	64.98	73.52	73.42	79.22	85.4	84.44	101.4	120	187	276.4	329.8
0.75	79.74	73.04	75.04	86.1	101.7	93.92	79.9	115.4	178.6	175.8	260.9
0.80	93.66	133.3	131.7	134.7	130.1	140.5	158.6	186.1	218.9	351	434
0.85	107.3	106.1	134.4	137.2	117.4	157	130.9	137.3	156	310.7	352.6
0.90	181.9	181.9	147	164	205.9	232.9	195	189.9	235.4	250.7	308.4

Table 12.3 Percentage reduction of the DP against the fixed-time one.

Junction A	Junction B										
	0.45	**0.50**	**0.55**	**0.60**	**0.65**	**0.70**	**0.75**	**0.80**	**0.85**	**0.90**	**0.95**
0.40	30.69	28.17	24.79	21.06	12.98	5.45	14.06	32.43	70.05	72.74	88.65
0.45	-18.5	23.13	14.12	13.64	3.19	-9.8	5.41	5.7	27.27	21.09	55.44
0.50	17.16	-30.6	19	8.94	-8.6	-10.6	-1.16	-3.09	41.29	43.29	36.93
0.55	8.1	9.44	-51.7	22.71	47.48	36	36.93	80.16	85.2	85.32	89.05
0.60	-2.84	-4.5	-6	-65.6	16.31	17.98	65.2	36.32	80.6	80.15	82.44
0.65	-11.1	-3.88	-13.7	-16.9	-58.3	5.34	57.94	31.48	41.78	72.39	78.66
0.70	-7.58	-14.9	-6.1	-7.64	-10.2	-65.9	8.94	60.84	70.2	44.19	54.95
0.75	-4.37	8.24	-0.19	-5.39	-5.61	19.66	-33.6	21.52	-10.4	56.44	64.27
0.80	-36.7	-33.5	-56.7	-13.5	38.1	25.93	57.67	-158	-34.3	-69.7	-183
0.85	-16.3	-2.51	-53.5	-1.24	36	35.17	70.55	75.77	-95.2	37.29	49.5
0.90	-18.2	-18.2	12.93	11.46	25.21	39.94	58.33	72.63	37.01	-27.5	58.63

As shown in Tables 12.1 and 12.2, when 2 junctions having equal flow-rates (i.e. in the case when both junctions having common cycle-length), the fixed-time control performs better than the proposed controller does. When two junctions having common cycle-length, the fixed-time controller provides optimal control, especially along the priority route. Since both junctions share the same starting and ending times in the priority route, vehicles traveling from junction B to A using the priority route are discharged completely in junction B without making any stops. As a result, no delay will be imposed on the vehicles coming from priority route of junction A when the fixed-time controller is optimal.

However, this situation is quite different when the 2 junctions do not have common cycle time. Without common cycle time, the 2 junctions no longer share common starting and ending time on the priority route, which results in poorer performance. For example, the cost z of the fixed-time control is 49.8 when the flow-rates of both junctions are 0.65 veh/s. Then z rises to 77.5 when the flow-rate of junction A is increased slightly to 0.7 veh/s. The fixed-time control is very sensitive to the changes of flow-rate. It is the very reason why engineers always group multiple junctions together and force them to share common cycle time. However, the incompatibility of the cycle time over the boundary of junction groups decreases the performance of the traffic control. Hence traffic congestion always occurs in the boundary of junction groups.

When the flow-rates of junction B is greater than that of junction A, the decentralized control performs much better than the fixed-time control. It is because when the flow-rate of junction B is very large, the right-to-left direction of the common link will become saturated very quickly. The DP coordination module located at junction A can foresee the problem and discharge the common link quickly before it gets saturated. The fixed-time control, however, does not have this capability. As a result, the DP decentralized controller performs generally better in the cases when flow-rates at junction B are larger. However, there are some cases that z of the fixed-time is better, such as when flow-rates of junction A and B are 0.5 veh/s and 0.7 veh/s respectively. This phenomenon is mainly due to the difference in the cost function used in system performance evaluation and the one used in DP. Equation 12.1 allows a strong contribution on the priority route. For example, the average delay in Junction A, B and the priority route delay for the fixed-time is 36, 67 and 74.5 s/veh respectively, while those for the DP are 35, 63 and 90.12 s/veh respectively. As seen in the figures, the average delay in individual junction of the DP is less than that in the fixed-time. In other words, the decentralized control is very good for local optimization, but it does not guarantee to reach the global optimal.

In the cases when the flow-rates of junction A are greater than that of junction B. Since the flow-rates of the traffic streams coming from junction B is not large when compared with the others, the dynamic programming controller in junction A will try to maximize junction usage of the other traffic streams. It is very similar to the case discussed above when the flow-rates of junction A and B are 0.5 and 0.7 veh/s. These cases show the local junction optimal is different from the global network optimal. This is the drawback of applying decentralized approach. Although the results obtained in decentralized approach may not be the global optimal, but it is still near to optimal.

12.4 Time varying Traffic Condition

Under time varying traffic condition testing, the overall flow-rates profile is similar to

Figure 11.3 and the junction layout is the same as Figure 12.1. Two junctions A and B resemble the boundary situation of 2 different junction groups. Junction A is connected to city center while junction B is connected to residential area. During "MP", majority of vehicles is coming from junction B and the direction reverses in "EP". Flow-rates of 2 junctions under "NT" and "DT" are set equal, while the 2 peak periods are not. In "MP", flow-rates of junction A are greater than that of junction B while in "EP", the flow-rates of junction A is smaller than that of junction B. Turning percentage is set to 5% under "NT" and "DT", and 10% under 2 peaks. Result is shown in Figure 12.2 and Table 12.4.

The cost z is calculated using equation 12.1 where no priority route set in "NT" and "DT". The priority route of "MP" is set to traffic stream coming from junction B to junction A, while the priority route of "EP" is set to traffic stream coming from junction A to junction B. The smaller the z, the better the system is.

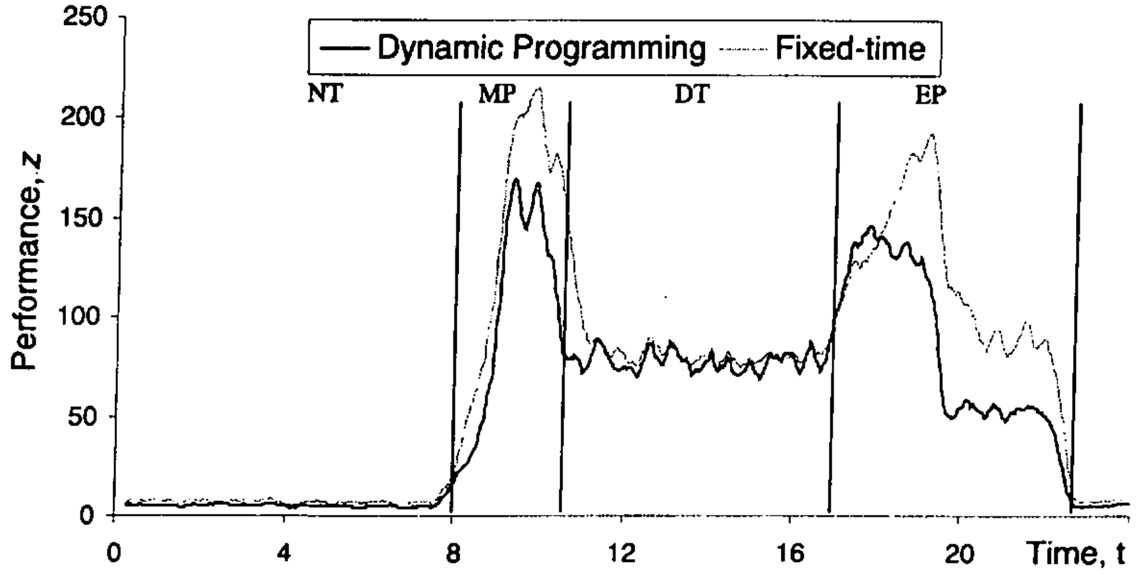

Figure 12.2 Cost z of DP and fixed-time coordination control under time varying traffic condition.

Table 12.4 Cost z of DP and fixed-time control in different time zone.

Time Period	Fixed-time	Dynamic Programming	Reduction %
NT	25.49	27.69	-8.63
MP	147.30	106.12	27.96
DT	77.95	83.79	-7.49
EP	115.46	84.70	26.64
Overall	65.25	59.89	8.21

As shown in Table 12.4, under NT and DT, the fixed-time control is better than the decentralized one. This is because under these time periods, the flow-rates of 2 junctions are similar. This means the 2 junctions share common cycle length during these 2 periods. From the results obtained in last section, when 2 junctions having the same cycle length, the fixed-time control will becomes optimal. The decentralized control can still produce comparative results with only 8% worse.

The decentralized control performs best under time-varying condition where the flow-rates are different, especially under the peak periods. It achieves more than 20% reduction in the performance index. This is because under the 2 peak periods, the junctions do not share a common cycle length and hence the fixed-time control is no longer optimal. The dynamic programming coordination module of the decentralized control optimizes the junction and produce significant results. On average, the decentralized control improves the system by around 8%.

12.5 Conclusions

The dynamic programming coordination module enables real time optimization and selects the best control action under all possible cases. The result obtained may not be optimal for the whole network but it is still near enough to the optimal one. Unlike the fixed-time control, the decentralized control does produce a satisfactory result over a wide range of traffic conditions. The controller is best placed in area where the flow-rates of the junctions are different, such as in the junction having 2 or more major traffic streams intersect.

The dynamic programming control module together with the HFLC constructed in the previous chapters form a decentralized junction controller. The encouraging simulation results show that the controller is capable to handle time-varying traffic. Last but not least, the relative simplicity in structure and the moderate computational demand allows the controller to be realized by any ordinary personal computer. As a result, a high-cost centralized controlling computer can be replaced by many low-cost decentralized controller located in every single junction.

13 Conclusions

A generic decentralized traffic junction controller using artificial intelligence is proposed. The controller is simple in structure and does not require any geographic knowledge of the junctions. Each junction has a decentralized traffic junction controller that can be connected together to form traffic network control. Unlike the centralized control method that is also in common use, the proposed controller does not impose any limit on the number of junctions in the network. The size of the network does not affect the performance of the proposed controller. In fact, unlimited number of controllers can join together to form an infinitely large network.

With the help of vehicle detectors, the proposed controller responds very quickly to the changes of traffic condition. The architecture of the decentralized controller speeds up the response time to the traffic. Unlike centralized control, each decentralized controller in the network only needs to react with the local traffic only. The amount of data needs to be processed for the decentralized controller is much less than those in the centralized one. As a result, shorter calculation time and hence faster decision can be made. Moreover, the proposed controller does not require any expensive hardware.

The proposed controller consists of 3 modules, a local decision module, an optimization module and a junction coordination module. At each signal phase, the local decision module based on data obtained from the vehicle detectors and makes decision on how long this phase will last. The decision is made on local data only, without any knowledge of historical data and any nearby junction information. The local decision module always works as a single isolated junction controller. The optimization module provides explicit knowledge of the control parameters to the local decision module. The optimization module is an offline module that can be turned on at any time to provide fine-tuning of the local decision module. Moreover, the knowledge obtained from a single controller can be shared within the network. That is, optimization process can be performed at a single junction and the result can be deployed to others. The junction coordination module groups junctions together to form a controlled network. It communicates with the adjacent decentralized controller to produce coordinated results.

The local decision module is a fuzzy controller with a hierarchical structure. It divides the 3-input 1-output fuzzy system into two 2-input 1-output systems. It reduces the total number of fuzzy rules from N^3 to $2N^2$, where N is the number of fuzzy labels and in this case is equal to 16.

The optimization module use Genetic Algorithms to produce fuzzy rules for the local decision module. It replaces the trial-and-error rule-definition process by an automatic learning. Moreover, the GA is capable of producing a minimum set of rules necessary to the system. All unnecessary rules are removed by the natural selection function of GA. With the help of the optimization module, the local decision module future reduces the local average delay and a local optimal is achieved.

The junction coordination module adopts real time dynamic programming to obtain coordinated result. Using the coordination parameters and the result of the local decision module, the coordination module calculates the effective green time of a signal phase that can achieve network optimal. The coordination parameters are given from adjacent junctions, which reflect the incoming flow-rate of that junction in the near future. Given the near future incoming flow-rate, the module can adjust the local effective green time that adapt to adjacent junction situation.

For single junction control, the proposed controller improves more than 3 to 24% in general. For a small network of 2 junctions, the proposed controller is 8% worse if 2 junctions have common cycle time. In the case when 2 junctions do not have the same cycle time, the proposed controller improves the system by around 8%. As a result, the proposed controller cannot provide optimal control if the junctions have the same flow-rates. Nevertheless, the controller can still produce near optimal control.

14 Further Improvements

There are few things that can be improved later. Recall the memory again: the HFLC being built consists of 3 levels. The first level, local decision module, is a fuzzy controller having pre-defined rules. The second level, optimization module, is a fuzzy controller having rules generated by static GA learning. The third level, junction coordination module, is a dynamic programming with a fixed objective function.

14.1 Road Pricing Scheme

As mentioned in the synopsis, the objective of this study is to construct a generic controller that capable to traffic in a microscopic viewpoint. The study does not cover the integration to the macroscopic policy such as road pricing scheme. If road pricing is introduced, it will affect driver behavior. The flow-rate will drop suddenly at the initial stage and then increase gently. It is the same case as increasing toll fee of a paid-road system. The HFLC is capable to handle the changes in the flow-rate without making any modification. However, the policy planner may like to give more privilege to the paid-road users such as much faster access. Then the junction coordination module will need some enhancement. The enhancement will be discussed shortly in session 14.4. The basic idea is to adjust the objective function such that more weight can be put into the privilege traffic stream.

14.2 Local Decision Module

As the rules are generated using equation 5.3, there is no much that can be improved, unless a more accurate equation can be defined later.

14.3 Optimization Module

The performance of the controller is mainly based on the "quality" of the rules. The "quality" of the rules in this level can be improved by changing the static learning method of GA into continuous one. Continuous learning means that real time data is feed into the GA learning engine. The learning engine will handle the creation and removal of the KB rules in the real time. As a result, the controller will be able to adapt itself for the changes of traffic pattern. Chapter 8 mentions that a rule in the KB will not be removed unless a contradictory one is generated. This means that a rule having very low activity (rule firing rate) can stay in the KB. By introducing the "number of fires" and "gene age" into the KB, those rules having low activity can be removed. Hence a compact and efficient KB can be maintained.

14.4 Junction Coordination Module

Currently only a standard objective function is applied in the dynamic programming. This objective function treats all traffic streams equally and gives out an average result. But under certain situations, the traffic streams cannot be treated as equally weighted. For example, some streams are defined as priority streams in the macroscopic sense. More green time should be assigned for the high priority streams. As a result, the objective function should be changed to a weighted sum and let the designer assign the weights.

14.5 Hybrid Approach

Examinations in chapter 12 showed that there exist some cases that the fixed-time control methodology performs better than the HFLC especially under flow-rate of 2 junctions are equal. In order to enhance the controller performance, a hybrid approach control method can be used. A threshold is first defined. If the flow-rate difference between 2 adjacent junctions is less than the threshold, the fixed-time control methodology is chosen. Otherwise, the HFLC is used. The hybrid approach has the benefit of both the fixed-time and HFLC. For example, under conditions when both junctions having the same cycle time, the fixed-time controller can gives the optimal result. When both junctions do not share the common cycle time, the HFLC performs best. The threshold is used in order to prevent too often switching process as the flow-rates may vary slightly.

References

1. Al-Khalili, A.J. and Macleod, C.J. "Optimum Offset - I: Determination of the optimum offset between two adjacent road junctions using a hill-climbing technique". *Transportation Research* (1975)
2. Al-Khalili, A.J. and Macleod, C.J. "Optimum Offset - II: Determination of the optimum offset between two adjacent road junctions using a regression analysis". *Transportation Research* (1975)
3. Al-Khalili, A.J. "Urban Traffic Control – A General Approach". *IEEE Transaction on Systems, Man and Cybernetics* (1985)
4. "The Annual Traffic Census". *Transport Department*, Hong Kong Special Administrative Region Government
5. Berenji, H.R. "Learning and tuning fuzzy logic controllers through reinforcements". *IEEE Trans. Neural Networks*, Vol. 3, pp. 724-740 (1990)
6. Chiu, S. and Chand, S. "Adaptive Traffic Signal Control Using Fuzzy Logic". *IEEE Proceedings of Intelligent Vehicles '92 Symposium*, pp. 101-106 (1992)
7. Crabtree, M.R., Vincent, R.A. and Harrison, S. "TRANSYT/10 User Guide". *TRL Application Guide 28*, Transport Research Laboratory, Crowthorne (1996)
8. Daley, S. and Gill, K.F. "A design study of a self-organising controller". *Proc. Inst. Mech. Eng. C*, Vol. 200, pp. 59-69 (1986)
9. Elahi, S.M., Radwan, A.E. and Goul, K.M. "Traffic Signal using Mixed Controller Operations". *Journal of Transportation Engineering*, Vol. 118 (6), pp. 866-881 (1992)
10. Findler, N.V. and Stapp, J. "Distributed Approach to Optimized Control of Street Traffic Signals". *Journal of Transportation Engineering*, Vol. 118 (1), pp. 99-110 (1992)
11. Findler, N.V., Surender, S., Ma, Z. and Catrava, S. "Distributed Intelligent Control of Street and Highway Ramp Traffic Signals". *Engineering Application Artificial Intelligent*, Vol. 19 (3), pp. 281-292 (1997)
12. Glover, F. "Tabu search-part I". *ORSA J. Comput.*, Vol. 1 (3), pp. 190-206 (1989)
13. Glover, F. "Tabu search-part II". *ORSA J. Comput.*, Vol. 2 (1), pp. 4-32 (1990)
14. Goldberg, D.E. "Genetic Algorithms in Search, Optimization and Machine Learning". *Addison Wesley* (1989)

15. Heydecker, B.G. "A Decomposition Approach for Signal Optimization in Road Networks". *Transpn. Res. B*, Vol. 30 (2), pp. 99-114 (1996)

16. Ho, F.S. and Ioannou, P. "Traffic Flow Modelnig and Control Using Artificial Neural Networks". *IEEE Control Systems*, pp. 16-26 (1996)

17. Ho, T.K. "Fuzzy logic traffic control at a road junction with time-varying flow rates". *Electronics letters*, Vol. 32 (17), pp. 1625-1626 (1996)

18. Holland, J.H. "Adaptation in Natural and Artificial Systems". *MIT press* (1975)

19. Ishibuchi, H., Yamamoto, N., Murata, T. and Tanaka, H. "Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms". *Fuzzy Sets System*, Vol. 65, pp. 237-253 (1994)

20. Ishibuchi, H., Yamamoto, N., Murata, T. and Tanaka, H. "Genetic algorithms and neighborhood search algorithms for fuzzy flow-shop scheduling problems". *Fuzzy Sets System*, Vol. 67, pp. 81-100 (1994)

21. Karr, C.L., Belew, R. and Booker, L. "Genetic Algorithm based Fuzzy Control of Spacecraft Autonomous Rendezvous". *Proceedings of international conference on Genetic algorithms* (1991)

22. Karr, C.L. and Gentry, E.J. "Fuzzy Control of pH using Genetic Algorithms". *IEEE Trans. Fuzzy System*, Vol. 1 (1), pp. 46-53 (1993)

23. Karr, C.L. and Schaffer, J. "Design of an adaptive fuzzy logic using a genetic algorithm". *Proceedings of international conference on Genetic algorithms* (1989)

24. Lee, C.C. "Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I". *IEEE Transaction On Systems, Man, and Cybernetics*, Vol. 20, pp. 404-418 (1990)

25. Lee, C.H. and Wang, S.D. "A self-organizing adaptive fuzzy controller". *Fuzzy Sets and Systems*, Vol. 80, pp. 295-313 (1996)

26. Linkens, D.A. and Abbod, M.F. "Self-organizing fuzzy logic controllers for real-time processes". *Proceedings of IEE Control 91*, pp.971-976 (1991)

27. Luger, G.F. and Stubblefield, W.A. "Artificial Intelligence: Structures and Strategies for Complex Problem Solving", *Addison Wesley* (1993)

28. Luk, J.Y.K. "Two traffic-responsive Area Traffic Control methods: SCAT and SCOOT". *Traffic Engineering + Control* (1984)

29. Meredith, D.L., Karr, C.L. and Kumar, K.K. "The use of Genetic Algorithms in the design of Fuzzy Logic Controllers". *Third workshop on Neural networks WNN92*, pp. 549-555 (1993)
30. Pappis, C.P. and Mamdani, E.H. "A Fuzzy Logic Controller for a Traffic Junction". *IEEE Transactions on Systems, Man, and Cybernetics* (1977)
31. Procyk, T.J. and Mamdani, E.H. "A linguistic self-organizing process controller". *Automatica*, Vol. 15, pp. 15-30 (1977)
32. Rach, L. "The development and evaluation of metropolitan Toronto's real-time program for computerized traffic control devices". *Proc. IFAC/IFIP/IFORS 3rd Int. Symp. On Control in Transportation Systems*, Ohio, pp. 349-362 (1976)
33. Robertson, D. and Wilson, N. "The Effects of Coordinated and Isolated Signal Control on Journey Times and Exhaust Emissions along the A12 in London". *Traffic Engineering + Control*, Vol. 1, pp. 4-9 (1996)
34. Robertson, D.I. "TRANSYT: A traffic network study tool". *Department of the Environment, RRL Report LT 253* (1969)
35. Ross, D.W., Humphrey, E.E., Mahoney, R.C., Sandys, R.C., Williams, G.L., Whonoutka, R., Wong, P.J. and Leidler, H.M. "Improved control logic for use with computer-controlled traffic". *NCHRP Project 3-18(1) Report*, Stanford Research Inst., California (1977)
36. Shibata, T. and Fukuda, T. "Coordinative behavior by genetic algorithm and fuzzy in evolutionary multi-agent system". *Proceedings of IEEE International Conference on Robotics and Automation*, Atlanta, USA, pp. 760-765 (1993)
37. Spall, J.C. and Chin, D.C. "A model-free approach to optimal signal light timing for system-wide traffic control". *Proceedings of the 33rd Conference on Decision and Control*, pp. 1868-1875 (1994)
38. Sugeno, M. and Nishida, M. "Fuzzy control of model car". *Fuzzy Sets System*, Vol. 16, pp. 103-113 (1985)
39. Takagi, T. and Sugeno, M. "Derivation of fuzzy control rules from human operator's control actions". *Proceeding of the IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis*, Marseilles, France, July, pp. 55-60 (1983)

40. Tarng, Y.S., Yeh, Z.M. and Nian, C.Y. "Genetic synthesis of fuzzy logic controllers in turning". *Fuzzy Sets and Systems*, Vol. 83, pp. 301-310 (1996)

41. Tarnoff, P.J. "The results of FHWA urban traffic control research: an interim report". *Traffic Engineering*, Vol. 45 (4), pp. 27-35 (1975)

42. Webster, F.V. "Traffic Signal Setting". *Technical paper 39*, Road Research Laboratory (1958)

43. Zadeh, L.A. "Fuzzy Sets". *Information Control*, Vol. 8, pp. 338-353 (1965)

Index to Figures

Index to Tables

Index to Equations