TRAFFIC SIGNAL CONTROL WITH SWAM INTELLIGENCE
ANT COLONY OPTIMIZATION

A Thesis
presented to
the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical Engineering

by
Pang-shi Shih
June 2013

© 2013
Pang-shi Shih
ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: Traffic signal control with swam intelligence ant colony optimization

AUTHOR: Pang-shi Shih

DATE SUBMITTED: June 2013

COMMITTEE CHAIR: Dr. Xiao-Hua (Helen) Yu, Professor of Electrical Engineering

COMMITTEE MEMBER: Dr. Taufik, Professor of Electrical Engineering

COMMITTEE MEMBER: Dr. Xiaomin Jin, Associate Professor of Electrical Engineering

ABSTRACT

Traffic signal control with swarm intelligence ant colony optimization

Pang-shi Shih

Ant colony optimization (ACO) is a meta-heuristic based on the indirect communication of a colony of artificial ants mediated by pheromone trails with collaboration and knowledge-sharing mechanism during their food-seeking process. ACO has been successfully applied to solve many NP-hard combinatorial optimization problems such as travel salesman problem, quadratic problem, just to name a few. In this research, we apply the ACO algorithm to the traffic signal control in order to minimize the user delay at a traffic intersection. Simulation results from our computational experiments indicate that ACO provides better performance during high traffic demand, compared to the conventional Fully Actuated Control (FAC).

Keywords: Ant colony optimization (ACO), meta-heuristic, the traffic signal control, user delay

TABLE OF CONTENTS

LIST OF TABLES	vii
LIST OF FIGURES	viii
Chapter 1 Introduction.....	1
Chapter 2 Literature Review.....	3
Chapter 3 Traffic Dynamics	6
3.1 Problem Statement	6
3.2 Traffic Terminology.....	7
3.3 Arrivals	7
3.4 Departures	8
3.5 Traffic Flow Command.....	8
3.6 Vehicle Delays	9
3.6.1 Deterministic part: delay of vehicle initially in the queue (no new arrivals)	9
3.6.2 Probabilistic part: delay of new arrivals not released on current phase	10
3.6.3 Probabilistic part: delay of new arrivals released on current phase	11
3.7 Vehicle Delay During a Signal Phase	12
3.7.1 Case 1: The queue is empty at the end of the phase.....	12
3.7.2 Case 2: All initial vehicles are released but the queue is not empty at the end of the phase ...	13
3.7.3 Case 3: Not all initial vehicles are released	14
3.7.4 Case 4: On red movement	14
3.8 Total Vehicle Delay Time During a Signal Period	15
3.9 Dual-ring Traffic Signal Control.....	15
3.9.1 Phase numbers	16
3.9.2 Dual-ring control.....	16
Chapter 4 Ant Colony Optimization.....	19
4.1 Fundamentals of ACO	19
4.2 Ant Colony Optimization Framework and Flowchart	20
4.3 Variety of Ant Colony Optimization Algorithm.....	24
4.3.1 Ant system (AS).....	24

4.3.2	Elitist ant system (EAS)	25
4.3.3	Rank-based ant system (RBAS).....	25
4.4	Application of the ACO	26
Chapter 5	Application of Ant Colony Optimization and Traffic Signal Control	27
5.1	Inspiration / Motivation	27
5.2	Ant Colony Optimization Implementation	27
5.2.1	Rolling horizon control	27
5.2.2	The local search	32
5.2.3	Complete the entire signal period	33
5.3	Fully Actuated Control	33
Chapter 6	Computer Simulation Results	35
6.1	The Environment of the Pheromone Level Convergence to the Optimal Solution	35
6.2	Ant System (AS)	37
6.3	Rank-based Ant System (RBAS).....	42
6.4	The Average Delay	47
Chapter 7	Concluding Remarks and Future Research	56
References.....		58
Appendices		
Appendix A	Tables of Delay Time and Waiting Queues	61
Appendix B	Matlab Code.....	74

LIST OF TABLES

Table 6-1. Traffic Parameters	35
Table 6-2. ACO Parameters.....	37
Table 6-3. Delay time (ACO with $LT/T=1$).....	53
Table 6-4. Delay time (FAC with $LT/T=1$)	53
Table 6-5. Queue length (ACO with $LT/T=1$).....	53
Table 6-6. Queue length (FAC with $LT/T=1$).....	54
Table 6-7. Delay time (ACO with $LT/T=0.5$).....	54
Table 6-8. Delay time (FAC with $LT/T=0.5$)	54
Table 6-9. Queue length (ACO with $LT/T=0.5$).....	55
Table 6-10. Queue length (FAC with $LT/T=0.5$).....	55

LIST OF FIGURES

Figure 3-1. An isolated intersection with eight movements	6
Figure 3-2. The traffic signal phase	16
Figure 3-3. Eight-phase dual-ring signal control	17
Figure 3-4. The order of dual ring control	18
Figure 4-1. Behavior of real ants to find a shortest path by pheromone	20
Figure 4-2. ACO pseudo-code	22
Figure 4-3. ACO flowchart	23
Figure 5-1. Maximum time intervals of a signal cycle	28
Figure 5-2. Example of rolling horizon control	28
Figure 5-3. The candidate solutions for ants to travel.....	29
Figure 5-4. ACO computational flow chart	31
Figure 5-5. FAC computational flow chart.....	34
Figure 6-1. Ant System rate of convergence with 10 ants, $\rho=0.2$	38
Figure 6-2. Ant System rate of convergence with 25 ants, $\rho=0.2$	38
Figure 6-3. Ant System rate of convergence with 50 ants, $\rho=0.2$	39
Figure 6-4. Ant System average rate of convergence with $\rho=0.2$	39
Figure 6-5. Ant System rate of convergence with 10 ants, $\rho=0.4$	40
Figure 6-6. Ant System rate of convergence with 25 ants, $\rho=0.4$	40
Figure 6-7. Ant System rate of convergence with 50 ants, $\rho=0.4$	41
Figure 6-8. Ant System average rate of convergence with $\rho=0.4$	41
Figure 6-9. Rank-based Ant System rate of convergence with 10 ants, $\rho=0.2$	42
Figure 6-10. Rank-based Ant System rate of convergence with 25 ants, $\rho=0.2$	43
Figure 6-11. Rank-based Ant System rate of convergence with 50 ants, $\rho=0.2$	43
Figure 6-12. Rank-based Ant System rate of convergence with $\rho=0.2$	44
Figure 6-13. Rank-based Ant System rate of convergence with 10 ants, $\rho=0.4$	44
Figure 6-14. Rank-based Ant System rate of convergence with 25 ants, $\rho=0.4$	45
Figure 6-15. Rank-based Ant System rate of convergence with 50 ants, $\rho=0.4$	45
Figure 6-16. Rank-based Ant System rate of convergence with $\rho=0.4$	46
Figure 6-17. Average delay for $(LT/T)=1$	48
Figure 6-18. Upper and lower bounds on average delay for $(LT/T)=1$	49
Figure 6-19. Average queue length for $(LT/T)=1$	49
Figure 6-20. Upper and lower bounds on queue length for $(LT/T)=1$	50
Figure 6-21. Average delay for $(LT/T)=0.5$	50
Figure 6-22. Upper and lower bounds on average delay for $(LT/T)=0.5$	51
Figure 6-23. Average queue length for $(LT/T)=0.5$	51
Figure 6-24. Upper and lower bounds on queue length for $(LT/T)=0.5$	52
Figure 7-1. A typical elementary traffic network with five intersections	56

Chapter 1

Introduction

Traffic signals have significant impacts on everyone's daily life. With the increased traffic demand, congestion has become a serious problem in many urban cities around the world. In general, efficiency and safety are the main considerations of the traffic control. By organizing the traffic demand at each intersection in the network, our objective is to avoid the traffic conflict and reduce traffic jam.

Many algorithms have developed for traffic signal control over the years. Some of the previous approaches include TRANSYT (Traffic Network Study Tool) [23], SCATS (Sydney Coordinated Adaptive Traffic System) [16], and SCOOT (Split Cycle and Offset Optimization technique) [12]. More modern schemes such as using Markov Decision Process (MDPs) [26] and fuzzy logic [17] also have been investigated in recent years.

Swarm intelligence is a relatively new approach to traffic signal control. Ant colony optimization (ACO) algorithm is a meta-heuristic for solving computationally hard combinatorial problem [3] [9]. Ant colony optimization takes inspiration from the foraging behavior of some ant species. These ants deposit pheromone on the ground when searching for food in order to mark some favorable paths that should be followed by other member of the colony. As more pheromone is deposited, the paths become less random and are biased toward to the paths with higher pheromone concentration. In ACO algorithm, each single agent is called an artificial ant. All artificial ants search the solution space in a probabilistic manner to create candidate solutions. These candidate solutions are then evaluated based on the amount of the pheromone concentrations. When time goes by, only the paths with large amount of pheromone are kept.

In this thesis, ACO algorithm with rolling horizon approach is employed to achieve real-time adaptive control in order to find the optimal traffic signal at an isolated intersection. Computer simulation results indicate that this new approach is more efficient than traditional Fully Actuated Control (FAC) algorithm, especially under high traffic demand.

This thesis is organized into seven chapters. In Chapter 2, we present a literature review. The traffic dynamics and problem definition will be presented in Chapter 3. In Chapter 4, we introduce the ACO algorithm and some previous applications. Chapter 5 is devoted to explain how ACO is implemented in the traffic signal control problem. Chapter 6 contains computer simulation results. Finally, concluding remarks and discussions are given in Chapter 7.

Chapter 2 Literature Review

Traffic congestion has become one of the major issues since the industrial and economic growth. The cost of traffic congestion keeps increasing rapidly; and more specific, as commute times have more than doubled in recent 30 years, and rush-hour delays cost the nation more than \$100 billion per year, or about \$750 for every U.S. commuter, according to the Urban Mobility Report released Tuesday by the Texas Transportation Institute [10]. The fact is the cost of congestion continues to affect nation's economic competitiveness and productivity, causes companies and consumers billions of dollars each year. Additionally, traffic congestion also causes non-recoverable damage such as air pollution, gasoline consumption, and car accidents. Therefore, in order to reduce traffic congestion, improving traffic signal control strategies is a very effective way to ameliorate traffic management.

Traffic control is a difficult problem because traffic flow is stochastic and non-linear, such that many traditional control techniques are not able to provide the satisfactory solutions. Besides, traffic condition can change quickly, so control tactics should be extremely responsive. Due to traffic networks grow up, finding the optimal strategy becomes a complex combinational problem and it is complicated to make a real-time implementation. Therefore, new traffic control methods need to be investigated.

For large network, traffic control can be assorted in centralized and decentralized control [26]. In order to implement traffic control on large network, the network is divided into small subsystems. In centralized control system, a central control can send control signals to each subsystem and achieve global optimization because of the high level communication between the

subsystems. However, it is computationally intensive and very sensitive to a defective central controller or broken communication links, and it is not responsive to traffic dynamics. Due to these limitations, recent research has more focused on decentralized control. In decentralized control, adjacent subsystems communicate for coordination purposes [27]. It is less computationally complex and more responsive to time-varying traffic dynamics, but the optimization is made on a local level, not global.

Fuzzy logic has been introduced to traffic control in recent years. Fuzzy logic control has been successfully applied to not only an isolated intersection but also multiple intersections [17]. Theoretically, fuzzy logic controllers generate membership functions between the inputs and outputs. Then the controllers select an output which is competent to all membership functions. In traffic control system, the inputs are the present state of the intersection, such as current waiting time, the length of the vehicle's queue, and arrival rate. On the other hand, the outputs are the traffic control standard, such as extension green times and optimal signal cycle time. The benefit of using fuzzy logic control is to allowed for different objectives to be optimized simultaneously by determining a minimum level of acceptability for each objective. However, the negative side is the difficulty in optimizing the fuzzy logic rules bases.

In this research, a new technique of swarm intelligence, Ant Colony Optimization (ACO), is employed for the adaptive control of an isolated intersection. ACO algorithm has been successfully applied to many computationally complex combinatorial optimization problems [3] [9]. In fact, ACO algorithm has already been applied to traffic related problems, such as vehicle routing problem [5], and traffic signal optimization for an isolated intersection with through

movements [19]. In this thesis, we expand the work in [19] to include left-turn movements at an isolated intersection.

Chapter 3 Traffic Dynamics

In this chapter, the traffic signal control problem and traffic terminology are presented. The assumptions on vehicle arrivals and departures, and traffic flows are discussed.

3.1 Problem Statement

Consider a traffic intersection with eight external approaches. The odd numbers indicate the left turn movements, and the even numbers are the straight forward movements. For simplicity, each movement consists of a single lane only. Video detectors are located at the intersection to count the queue lengths. The estimate on volume of traffic is assumed. All vehicles are identical and leave the intersection with the same speed.

The objective is to find out the appropriate traffic signal length (or equivalently signal switch time) that minimizes the average delay of the vehicles at the intersection.

Figure 3-1. An isolated intersection with eight movements

3.2 Traffic Terminology

As mentioned above, there are total eight movements in an intersection. The number of vehicles waiting on each movement is called the queue. The all red time is the length of the time when all movements have red signals, and it occurs between phase changes to avoid car accidents. The minimum distance between vehicles is measured in seconds and called the minimum headway (hw). A vehicle's departure time minus its arrival time is called a vehicle delay time.

3.3 Arrivals

It is assumed that vehicles arrive at an intersection randomly and follows the Poisson process [25]. The vehicle arrival time is generated by the equation:

$$at_2 = at_1 + hw - \left(\frac{1}{\lambda} - hw\right) \times \text{Log}(\gamma) \quad (3.1)$$

where at_2 is the next arrival time, and at_1 is the former arrival time. λ is the arrival rate in vehicles per hour per movement. hw is the headway between each vehicle. $\text{Log}(\cdot)$ is the natural logarithm function and γ is a uniformly distributed random number between 0 and 1 [18]. Besides, in the Poisson process, the expected number of new arrival vehicles in Δt seconds is $\lambda \Delta t$ [25].

3.4 Departures

At a given time, t , the queue length on movement i is denoted as $q^i(t)$. The number of vehicles initially in the queue that leave on movement i during a time interval (t_1, t_2) is denoted as $q^i_{out}(t_1, t_2)$. The time intervals correspond with signal phases. The output $q^i_{out}(t_1, t_2)$ is a function of signal choice and the queue length at t_1 [20].

$$q^i_{out}(t_1, t_2) = \begin{cases} \min[q^i(t_1), 1 + \text{Int}\left(\frac{t_2 - t_1}{hw}\right)] & \text{if } u(t_1, t_2) = \text{green} \\ 0 & \text{if } u(t_1, t_2) = \text{red} \end{cases} \quad (3.2)$$

where $u(t_1, t_2)$ is the signal choice and $\text{Int}(\cdot)$ gives the integer part of its argument. The output function states that when the signal turns green, the first car in the queue leaves immediately, and then each successive vehicle leaves the intersection hw seconds after the vehicle in front of it until all the vehicles are released or the traffic signal light changes to red.

3.5 Traffic Flow Command

Vehicles are released from the queue sequentially when the signal turns green for a movement. The first vehicle leaves immediately when the traffic light changes with no delay time. Afterward, the following each vehicle leaves hw seconds after the vehicle in front of it. If additional vehicles arrive before the queue is empty, they are added to the end of the queue. New arrivals also leave hw seconds after the vehicle before it. This entire process continues until either the queue is empty or the traffic signal light changes to red. If the queue is empty before the traffic light switches to red, then additional vehicles go through the interaction freely without

any delay time. At the end of the green phase, the traffic signal light is red on all movements, which is called all red time. When a movement has a red light signal, new arrival vehicles are added to the end of the queue and wait until the next green phase to be released.

3.6 Vehicle Delays

Ant Colony optimization (ACO) is used to determine the traffic control signal to minimize vehicle delay time at the intersection. To apply this method, the algorithm evaluates candidate signal period by computing the total expected delay of the signal period; and this computation consists of a deterministic and probabilistic part. Generally, the deterministic part is the delay time of the vehicles which are already in the queue, and the probabilistic part is the expected delay time of future arrival vehicles. In order to calculate the expected delay time, the arrival rate of new vehicle is assumed to be known as stated in the problem statement.

3.6.1 Deterministic part: delay of vehicle initially in the queue (no new arrivals)

Given a green phase of the length $(t_2 - t_1)$, $q_{out}^i = q_{out}^i(t_1, t_2)$ of the initial vehicles will be released on green movements. Let $n \leq q_{out}^i$ represents the position in the queue of a vehicle which will be released on the current queue. At the beginning of a green phase, the current delay time of the n^{th} vehicle in the queue is $t_1 - at_n$, where at_n is the arrival time of

the n^{th} vehicle. The additional delay time is $(n - 1)hw$ seconds when the traffic signal light turns green. Therefore, the total delay of q_{out}^i vehicle is:

$$\begin{aligned} \sum_{n=1}^{q_{out}^i} (\text{wait time of vehicle } n) &= \sum_{n=1}^{q_{out}^i} (n - 1)hw + (t_1 - at_n) \\ &= \sum_{n=1}^{q_{out}^i} (t_1 - at_n) + \left[\frac{q_{out}^i(q_{out}^i - 1)}{2} \right] hw \end{aligned} \quad (3.3)$$

3.6.2 Probabilistic part: delay of new arrivals not released on current phase

When new vehicles arrive at the intersection, they are added to the end of the queue. They will be released after all the vehicles in the initial queue are released. Otherwise, they must wait to be released until future green phase. The expected delay time for the vehicles released during the phase they arrive is computed differently than those are not. In this section, the delay time of new arrival vehicles which are not released on current phase is discussed.

First, let Δt denote the length of the phase which new vehicles arrive at the intersection but do not pass. This situation takes place on a red signal phase or an over-congested green signal phase that does not allow new arrivals to pass. As mentioned earlier in section 3.3, in a Poisson process, the expected number of new arrival vehicles in Δt seconds is $\lambda\Delta t$. And the expected delay time for each new arrival vehicle is $\frac{\Delta t}{2}$ [21]. Therefore, the expected delay time for new arrivals during an interval of length Δt is:

$$\lambda \Delta t \times \frac{\Delta t}{2} = \frac{\lambda \Delta t^2}{2} \quad (3.4)$$

3.6.3 Probabilistic part: delay of new arrivals released on current phase

Alternatively, when all new arrival vehicles pass the intersection on the current green phase, the delay time of a new arrival depends on its position in queue when that vehicle arrives. This situation is similar to the delay of the vehicles initially in the queue at the beginning of a green phase. When a vehicle arrives, its delay time is the minimum headway (hw) multiplied by number of cars in front of it in the queue. Consequently, future expected queue lengths are necessary to calculate the expected delay time of future arrival vehicles.

According to the first new vehicle's arrival time, it can wait up to $(q - 1)hw$ seconds before it reaches to the front of the queue. Therefore, the average time to the front of the queue is:

$$\frac{\sum_{n=1}^q (n-1)hw}{q} = \frac{(q-1)hw}{2} \quad (3.5)$$

Equation (3.5) only provides the average wait time of each new arrival vehicle, but the expected number of new arrivals (before the queue is empty) must also be determined. This number is done by computing the expected time to an empty queue and then multiplying this time by the vehicle arrival rate λ , which is given by [20] :

$$\frac{(q-1)hw}{1-\lambda hw} \times \lambda \quad (3.6)$$

3.7 Vehicle Delay During a Signal Phase

The total expected delay time for a signal period is the sum over all eight movements, and it is computed by using equations (3.3), (3.4), (3.5), and (3.6). Because only one movement is considered at a time, notation can be simplified by letting $q = q^i(t_1)$ be the number of the vehicles for movement i at time t_1 .

In order to calculate the total expected delay time on a green phase, there are four different cases need to be considered [20]. The first case is if $t_2 - t_1 \geq \frac{(q-1)hw}{1-\lambda hw}$, all vehicles can be released. The second case is if $\frac{(q-1)hw}{1-\lambda hw} > t_2 - t_1 \geq (q-1)hw$, where all vehicles initially in the queue are released but the queue is not empty at the end of the phase due to new arrivals. The third case is if $(q-1)hw > t_2 - t_1$, where not all vehicles initially in the queue can be released. The last case is the expected delay time on the red movements.

3.7.1 Case 1: The queue is empty at the end of the phase

If $t_2 - t_1 \geq \frac{(q-1)hw}{1-\lambda hw}$, then there is no vehicles in the queue at the end of the phase. If the queue is initially empty, then the delay time for the movement during this phase will be zero. Otherwise, the total expected delay time from t_1 to t_2 is:

$$J_{green}(t_1, t_2) = \frac{q(q-1)hw}{2} + \frac{(q-1)hw}{2} \frac{(q-1)hw}{1-\lambda hw} \times \lambda \quad (3.7)$$

The first term is the delay of the initial vehicles equation (3.3). The second term is the average wait of future vehicles equation (3.5) multiplied by expected number of vehicles that will arrive before the queue is empty equation (3.6). When all vehicles are released from the queue, new arrivals pass through this intersection with no delay time.

3.7.2 Case 2: All initial vehicles are released but the queue is not empty at the end of the phase

If $\frac{(q-1)hw}{1-\lambda hw} > t_2 - t_1 \geq (q-1)hw$, then all initial vehicles are released. The number of vehicles that are released is $1 + \text{Int}\left(\frac{t_2-t_1}{hw}\right)$, and the number of vehicles that are expected to arrive is $\lambda(t_2 - t_1)$. Therefore, $q + \lambda(t_2 - t_1) - [1 + \text{Int}\left(\frac{t_2-t_1}{hw}\right)]$ vehicles are expected at the end of the phase, and the number of new arrivals released is:

$$rel = 1 + \text{Int}\left(\frac{t_2-t_1}{hw}\right) - q \quad (3.8)$$

The total expected delay time from t_1 to t_2 is:

$$J_{green}(t_1, t_2) = \frac{q(q-1)hw}{2} + \frac{(q-1)hw}{2} rel + \frac{\lambda(t_2-t_1-rel \times hw)^2}{2} \quad (3.9)$$

The first term is the delay of the initial vehicles equation (3.3). The second term is the average wait of future vehicles equation (3.5) multiplied by expected number of vehicles that will arrive and be released equation (3.8). The last term is the delay time of the vehicles that are not released equation (3.4).

3.7.3 Case 3: Not all initial vehicles are released

If $(q - 1)hw > t_2 - t_1$, where not all initial vehicles in the queue are released. Then $q_{out} = 1 + \text{Int}\left(\frac{t_2 - t_1}{hw}\right)$ vehicles will be released. During this phase, an additional $\lambda(t_2 - t_1)$ vehicles are expected to arrive and none of them are released. Therefore, $q + \lambda(t_2 - t_1) - [1 + \text{Int}\left(\frac{t_2 - t_1}{hw}\right)]$ vehicles are expected at the end of the phase.

The total expected delay time from t_1 to t_2 is:

$$J_{green}(t_1, t_2) = \frac{q_{out}(q_{out}-1)hw}{2} + (q - q_{out})(t_2 - t_1) + \frac{\lambda(t_2 - t_1)^2}{2} \quad (3.10)$$

The first term is the delay of the released vehicles equation (3.3). The second term is the delay of initial vehicles not released. The last term is the expected delay of future arrivals equation (3.4).

3.7.4 Case 4: On red movement

The total expected delay time also includes the delay on the red movements. During a red phase, there are no vehicles released and an additional $\lambda(t_2 - t_1)$ vehicles are expected to arrive. Therefore, $q + \lambda(t_2 - t_1)$ vehicles are expected at the end of the red phase.

The total expected delay time from t_1 to t_2 is:

$$J_{red}(t_1, t_2) = q(t_2 - t_1) + \frac{\lambda(t_2 - t_1)^2}{2} \quad (3.11)$$

The first term is the delay of the initial vehicles and the second term is the expected delay of future arrivals equation (3.4).

3.8 Total Vehicle Delay Time During a Signal Period

In order to reduce the delay of vehicles during the current period, it is important to ensure that future vehicles also have short delays. If the queue is too long at the end of a period, then the delay of future vehicles will be large. When evaluating candidate signal periods, the delay of vehicles left in the queue at the end of the signal period, t_3 , needs to be considered. Combining equations (3.7), (3.9), (3.10), and (3.11) the total expected delay time of a signal period from t_1 to t_3 is:

$$J(t_1, t_2, t_3) = \sum_{i=1}^8 \{ J_{u(t_1, t_2)}^i(t_1, t_2) + J_{u(t_2, t_3)}^i(t_2, t_3) + \sum_{k=1}^{q^i(t_1)} (t_1 - at_k^i) \} \quad (3.12)$$

where u denotes the signal choice on the movement. At any given t_1 , one set of movements will be green and the rest are red.

3.9 Dual-ring Traffic Signal Control

As mentioned earlier, our objective is to figure out the traffic signal length that minimizes the average delay of the vehicles at the intersection. A traffic control problem can be considered as a decision-making problem for a stochastic dynamic system. The dual-ring traffic signal control determines the sequence of traffic signals at an intersection [24].

3.9.1 Phase numbers

Phase number is assigned to each individual movement at the intersection. An eight phase dual-ring controller is shown in Figure 3-2.

Figure 3-2. The traffic signal phase

3.9.2 Dual-ring control

A barrier divides the eight different movements into two interlocked groups (rings): east/west and north/south. In each ring, four movements (two through movements and their corresponding left turn movements) must be served if there is demand. Theoretically, there are 48 different phase sequences available; however, in order to avoid conflict traffic, there are only

ten specific combinations are allowed in each ring. For example, on the left ring (east/west), there are three possible sequences if the initial signal is (1+5), which are $(1+5) \rightarrow (2+5) \rightarrow (2+6)$, or $(1+5) \rightarrow (1+6) \rightarrow (2+6)$, or $(1+5) \rightarrow (2+6)$. Other than that, there is no sequence allowed. Similarly, starting from signal (2+6), there are also three possible sequences, which are $(2+6) \rightarrow (2+5) \rightarrow (1+5)$, or $(2+6) \rightarrow (1+6) \rightarrow (1+5)$, or $(2+6) \rightarrow (1+5)$. Note there are only two possible choices if the initial signal is (2+5), which are $(2+5) \rightarrow (1+5) \rightarrow (1+6)$, or $(2+5) \rightarrow (2+6) \rightarrow (1+6)$. According to the same pattern, there are also two possible choices available if the initial signal is (1+6), which are $(1+6) \rightarrow (1+5) \rightarrow (2+5)$, or $(1+6) \rightarrow (2+6) \rightarrow (2+5)$. The entire method is also applied to the right ring (north/south), and all the possible sequences are listed below in Figure 3-4.

Figure 3-3. Eight-phase dual-ring signal control

Ease/ West	North/ South
$(1+5) \rightarrow (2+5) \rightarrow (2+6)$	$(3+7) \rightarrow (4+7) \rightarrow (4+8)$
$(1+5) \rightarrow (1+6) \rightarrow (2+6)$	$(3+7) \rightarrow (3+8) \rightarrow (4+8)$
$(1+5) \rightarrow (2+6)$	$(3+7) \rightarrow (4+8)$
$(2+6) \rightarrow (2+5) \rightarrow (1+5)$	$(4+8) \rightarrow (4+7) \rightarrow (3+7)$
$(2+6) \rightarrow (1+6) \rightarrow (1+5)$	$(4+8) \rightarrow (3+8) \rightarrow (3+7)$
$(2+6) \rightarrow (1+5)$	$(4+8) \rightarrow (3+7)$
$(2+5) \rightarrow (2+6) \rightarrow (1+6)$	$(4+7) \rightarrow (4+8) \rightarrow (3+8)$
$(2+5) \rightarrow (1+5) \rightarrow (1+6)$	$(4+7) \rightarrow (3+7) \rightarrow (3+8)$
$(1+6) \rightarrow (1+5) \rightarrow (2+5)$	$(3+8) \rightarrow (3+7) \rightarrow (4+7)$
$(1+6) \rightarrow (2+6) \rightarrow (2+5)$	$(3+8) \rightarrow (4+8) \rightarrow (4+7)$

Figure 3-4. The order of dual ring control

Chapter 4 Ant Colony Optimization

In this chapter, Ant Colony Optimization (ACO) is described. ACO can be used to solve combinatorial NP-hard problems [9], such as Traveling Salesman Problem (TSP).

4.1 Fundamentals of ACO

The ACO algorithm was first introduced by Coloni, Dorigo and Maniezzo (1991) and the first Ant System (AS) was proposed by Dorigo in his PhD thesis (1992) [6]. The ACO is a meta-heuristic algorithm which utilizes the inspiration from real ant colonies behavior to find the shortest path from a food source to the nest by pheromone information [3]. When ants seek for food, they leave a kind of chemical compositions which is called pheromone. The more ants walk through the path, the more pheromone left on the trail. Then, the next ant will choose one path with a probability proportional to the amount of pheromone. However, the pheromone trail starts to evaporate when time goes by. The more time it takes for an ant to travel back and forth, the more pheromones have to disappear. On the other hand, the pheromones are accumulated on the shorter paths because more ants actually travel through them. Eventually, this positive feedback process will construct the shortest path from their nest to the food source.

Figure 4-1 shows an example to represent the pheromone trail mechanism of real ant behavior [11]. Figure 4-1 (a) shows that the ants carry food from food source F to their nest N . When a leaf becomes an obstacle on the path, the ants have to decide whether to turn left or right at point D , see Figure 4-1 (b). The first ant has an equal probability of choosing the direction. It

will randomly turn left or right to get the food. The same situation happens at point E in the opposite trail direction as shown in Figure 4-1 (c). Suppose that all ants walk at the same velocity and have the same evaporation rate, more pheromone will be left on the path $NDBEF$ than $NDCEF$ because path $NDBEF$ is shorter than path $NDCEF$. The next ants will have a higher probability of choosing path $NDBEF$. Finally all ants will walk through the same shorter path by the positive feedback effect (Figure 4-1 (d)).

Figure 4-1. Behavior of real ants to find a shortest path by pheromone [6]

4.2 Ant Colony Optimization Framework and Flowchart

The above behavior of self-organized real ants with the ability to find the shortest path has inspired the development of ACO. Artificial ants cooperate to come up with the solution by exchanging information via pheromone deposited on paths. Consequently, the real ants have the following key characteristic:

- 1) Real ants prefer to choose the trail with higher pheromone.
- 2) The shorter distance of a path, the more pheromone leave on it.

3) Real ants communicate indirectly via pheromone.

The ACO algorithm has the following important steps:

- 1) *Initialize Pheromone Values*: Set up the same pheromone value on each path. It provides the same possibility for each ant to choose the solution at first.
- 2) *Solution Construction*: In ACO, the problem is treated by simulating a number of artificial ants moving on a graph. Each ant moves to the next node by the following probability:

$$p_{ij} = \begin{cases} \frac{\tau_{ij}^\alpha \eta_{ij}^\beta}{\sum_{l \in N_i} \tau_{il}^\alpha \eta_{il}^\beta} & \text{if } j \in N_i \\ 0 & \text{if } j \notin N_i \end{cases} \quad (4.1)$$

where p_{ij} is the probability between node i and j , and N_i is the set of neighborhood of node i that an ant has not visited yet, which includes all possible nodes that an ant can move to when it is at node i . The pheromone value on edge (i, j) is τ_{ij} ; η_{ij} is a heuristic value. The values of parameters α and β can be determined by the application.

- 3) *Pheromone Updating*: Our purpose is to get better solution by each time, therefore, the pheromone must be updated in order to evaluate the quality of the solution. At the end of each iteration, the pheromone is updated by:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \Delta\tau_{ij} \quad \forall (i, j) \in A \quad (4.2)$$

where ρ is the evaporation coefficient (coefficient of decay), $0 \leq \rho < 1$; and $\Delta\tau_{ij}$ is the pheromone update algorithm.

- 4) *Stopping criterion*: The ACO process is iterated until the end condition is met. The stopping criterion of the ACO could be the number of iterations, or CPU time, or when a good solution is obtained.

```
procedure ACO algorithm for static combinatorial problems
  Set parameters, initialize pheromone trails
  while (termination condition not met) do
    ConstructSolutions
    ApplyLocalSearch      % optional
    UpdateTrails
  end
end
```

Figure 4-2. ACO pseudo-code

Figure 4-3. ACO flowchart

4.3 Variety of Ant Colony Optimization Algorithm

The first ant colony optimization algorithm is known as Ant System (AS) [9] in early nineties. After then, several other ACO algorithms have been proposed. In this section, we present the original Ant System, and two most successful variants: Elitist Ant System (EAS) and Rank-Based Ant System (RBAS) [9]. Each algorithm uses m ants to construct candidate solutions; and the initial pheromone deposit and solution construction steps are the same. The difference between these three specific algorithms is the way how pheromone is updated.

Each artificial ant begins at a start node and constructs a solution. The solution constructed by ant k is denoted by T^k . Each solution T^k is evaluated by a cost function, denoted by C^k , which is the objective function being minimized.

4.3.1 Ant system (AS)

In AS algorithm, at the end of each iteration, the pheromone on edge(i, j) is updated by the following formula:

$$\tau_{ij}(n+1) = (1 - \rho)\tau_{ij}(n) + \sum_{k=1}^m \Delta\tau_{ij}^k \quad \forall (i, j) \in A \quad (4.3)$$

where m is the total number of ants, $\rho \in (0, 1]$ is the evaporation rate, $\Delta\tau_{ij}^k$ is the quantity per unit of pheromone laid on edge(i, j) by k -th ant at the current iteration. Increment $\Delta\tau_{ij}^k$ is given by:

$$\Delta\tau_{ij}^k = \begin{cases} 1/C^k & \text{if } edge(i,j) \text{ belongs to } T^k \\ 0 & \text{otherwise} \end{cases} \quad (4.4)$$

4.3.2 Elitist ant system (EAS)

In EAS algorithm, we use elitist strategy to emphasize the best-so-far solution, denoted as T^{bs} . The pheromone on $edge(i, j)$ is updated by the following formula:

$$\tau_{ij}(n+1) = (1-\rho)\tau_{ij}(n) + \sum_{k=1}^m \Delta\tau_{ij}^k + e\Delta\tau_{ij}^{bs} \quad \forall (i, j) \in A \quad (4.5)$$

where $\Delta\tau_{ij}^k$ is defined the same as in the Ant System algorithm, e is the weight for best-so-far path and $\Delta\tau_{ij}^{bs}$ is given by:

$$\Delta\tau_{ij}^{bs} = \begin{cases} 1/C^{bs} & \text{if } edge(i,j) \text{ belongs to } T^{bs} \\ 0 & \text{otherwise} \end{cases} \quad (4.6)$$

where C^{bs} is the cost function of the best-so-far solution.

4.3.3 Rank-based ant system (RBAS)

In RBAS algorithm, we rank the solutions in increasing order before the pheromone is deposited. There are only $(w-1)$ best-ranked ants and the best-so-far ants are allowed to deposit pheromone. The r^{th} best ant is weighted by $\max\{w-r, 0\}$, and the best-so-far solution is weighted by w . The pheromone on $edge(i, j)$ is updated by the following formula:

$$\tau_{ij}(n+1) = (1-\rho)\tau_{ij}(n) + \sum_{r=1}^{w-1} (w-r)\Delta\tau_{ij}^r + w\Delta\tau_{ij}^{bs} \quad \forall (i, j) \in A \quad (4.7)$$

where $\Delta\tau_{ij}^r$ and $\Delta\tau_{ij}^{bs}$ are defined above. The Rank-Based Ant System combines the advantage of both Ant System (AS) and Elite Ant System (EAS). This rank-based update process can eliminate some bad solutions and converge to an optimal solution quickly [9].

4.4 Application of the ACO

The ACO has been successfully used to solve many discrete optimization problems, such as quadratic assignment problems [8], partitioning problems [13], vehicles routing problems [5], vertex cover problems [14], generalized spanning tree problems [15], telecommunication network problems [4], and image process problems [2].

The Traveling Salesman Problem (TSP) is the well known ACO application [7]. TSP is known as a NP-hard problem, which is about a salesman who wishes to find the shortest path that visits each customer's city exactly once from a starting location. Once an ant has visited every city, it updates the pheromone value on the edge it traveled. d_{ij} denotes the distance between city i and city j . τ_{ij} is the pheromone laid on edge (i, j) , and η_{ij} is the visibility of value regarding edge (i, j) defined as $1/d_{ij}$. ACO algorithm has been shown to find the optimal solution to the TSP in fewer iterations than other naturally inspired algorithms, such as genetic algorithms and simulated annealing [7].

Chapter 5 Application of Ant Colony Optimization and Traffic Signal Control

In this chapter, the details of ACO application for traffic signal control problem are described.

5.1 Inspiration / Motivation

ACO algorithm has successfully been applied to many computationally complex combinatorial problems, and it already has been used to solve other traffic related problems, such as Vehicle Routing Problem (VPR) [5], with positive results. Therefore, it is a good choice to consider ACO algorithm to solve the traffic signal optimization problem.

5.2 Ant Colony Optimization Implementation

5.2.1 Rolling horizon control

Based on the Dual-Ring control, there are up to six time intervals with starting time instant from t_1 until t_7 . The following is an example to demonstrate this process for east/west movements.

Figure 5-1. Maximum time intervals of a signal cycle

Figure 5-2. Example of rolling horizon control

At first, we randomly decide a sequence based on the Dual-Ring Control to setup green signal, for example from movements (1+5) to movements (2+5). The length of a green signal in a candidate solution is bounded between the predetermined minimum green time t_{min} and maximum green time t_{max} . Additionally, in order to make the set of candidate solutions finite,

time is discretized into one second time intervals. Under these conditions, a graphic is illustrated for the artificial ants to traverse in Figure 5-3. When an artificial ant is at time t , the set of admissible nodes, that it can move to the candidate is $\{ t + t_{min}, t + t_{min} + 1, t + t_{min} + 2, \dots, t + t_{max} \}$. All artificial ants start at the current time and then traverse right to a new node which represents the next signal switch time. This process creates a full signal period. At a given time t_1 , the set of candidate solutions is all the possible admissible combinations of the next two signal switch times, t_2 and t_3 .

Figure 5-3. The candidate solutions for ants to travel

After an artificial ant creates a signal period from t_1 to t_3 , the expected total delay time, $J(t_1, t_2, t_3)$, of the artificial ant's solution is computed using equation (3.12). However, $J(t_1, t_2, t_3)$ is not quite the cost function needs to be minimized. The reason is shorter time intervals tend to have smaller total expected delay time due to fewer vehicles enter the intersection during the period. Therefore, the expected delay time of fewer vehicles is being summed. Short period lengths are suboptimal when they create long queue lengths. In order to avoid long queues situation, the total expected delay time is divided by the length of the period multiplied by the traffic volume plus the initial vehicles number. This calls the expected average delay time per vehicle. This value, not the total expected delay time, is the cost function need to be minimized [19].

$$C^k = \frac{J^k(t_1, t_2, t_3)}{4(\lambda_1 + \lambda_2)(t_3 - t_1) + \sum_{i=1}^8 q^i(t_1)} \quad (5.1)$$

where λ_1 and λ_2 are the arrival rates of forward and left turn movements respectively. The next step is to update pheromone on the edge which ant k traversed. The flow chart of ACO is shown in Figure 5-4.

Figure 5-4. ACO computational flow chart

As mentioned earlier, an advantage of the ACO is its ability to incorporate the heuristic information with the solution space being searched. In the traffic signal problem, to release all waiting vehicles in the queue usually cause less delay time. For small queues, the optimal signal switch time is to release the current vehicles. For longer queues, the extra time is optimal because, with high probability, additional vehicles will arrive before all vehicles are released. This time is $t_1 + (q^g(t_1) - 1) \times hw$, where $q^g(t_1)$ is the length of the largest queue on the green movements at time t_1 . In order to bias the solution towards signal switch times near this time, the pheromone values are weighted by the heuristic value of [19]

$$\eta_{t_1 t_2} = \text{Exp}\left[-\frac{|(q^g(t_1)-1) \times hw - (t_2 - t_1)|}{c}\right] \quad (5.2)$$

where c is a positive constant. The value of c is chosen experimentally. The exponential function is used because it has a sharp peak at its maximum.

5.2.2 The local search

One problem with the above ACO process is its tendency to accumulate pheromone on near optimal solution. At initialization, all paths are set with equal pheromone. If one near optimal solution is randomly chosen more than any other path, then pheromone maybe accumulated rapidly on this near optimal solution. As a result, the real optimal path may be ignored and never be found. When using an ACO algorithm with the best-so-far ant, this stagnation becomes especially apparent. To avoid this stagnation, a search of the solutions near

the best-so-far solution can be added, and this step is called a local search. Local search can be performed in many different ways and is dependent on the problem being optimized.

In the traffic signal problem, a local search is utilized by replacing every n^{th} iteration of the random solution search. In this local search, the search space is replaced with a neighborhood of size T of the best-so-far solution. On a normal iteration, the possible solutions of next switch times are for an ant at time t is the set, $\{t + t_{min}, t + t_{min} + 1, t + t_{min} + 2, \dots, t + t_{max}\}$. However, in a local search, the search space is restricted to the set, $\{t + t_{bt} - T, t + t_{bt} - T + 1, t + t_{bt} - T + 2, \dots, t + t_{bt} + T\}$ where t_{bt} is the best-so-far signal switch time. If a better solution is found, it replaces the previous best-so-far solution. This process leads to less stagnation and faster convergence [19].

5.2.3 Complete the entire signal period

The example above is to demonstrate how to apply ACO algorithm to get the next signal switch times between t_1 and t_3 . To finish the entire signal period, we can just repeat the process starts with t_2 again to calculate the rest of the signal switch times individually. All the optimal solutions can be found in the end by using the same method.

5.3 Fully Actuated Control

In this research, the results of traditional Fully Actuated Control (FAC) is compared to the results of ACO algorithm. Similar to ACO, in Fully Actuated Control (FAC), the minimum green time, the maximum green time, and an extension time are usually pre-set. First, the

minimum green time is assigned to certain movements. If there is an arrival on this green movement, the controller will extend the green signal with extension time. This process can continue until the maximum green time is reached. On the other hand, if there is no vehicle arrives on a green movement during an extension period, the controller will check for vehicle arrivals in red light movements. If there is a vehicle, signal is switched and that movement is given the minimum green time first and the above procedure is repeated. Otherwise, the traffic signal remains the same [19].

Figure 5-5. FAC computational flow chart

Chapter 6 Computer Simulation Results

In this chapter, the computer simulation results of two different ant colony algorithms. One is the Ant System (AS) and the other is the Rank-Based Ant System (RBAS) are presented. The rates of pheromone convergence to the optimal path are examined by these two algorithms. Table 6-1 shows the traffic parameters applied in the simulations.

Parameter	Value
Minimum Green Time (sec)	5
Maximum Green Time (sec)	30
All Red Time (sec)	2
Headway/hw (sec)	2
Extension Time (sec)	1

Table 6-1. Traffic Parameters

6.1 The Environment of the Pheromone Level Convergence to the Optimal Solution

To estimate the pheromone level convergence rates, the pheromone concentration on the optimal path is recorded. If the algorithm converges, the pheromone becomes stabilized concentrated on a path. In order to see the qualitative behavior of the pheromone convergence, consider a simple case of zero initial condition (i.e., there is no vehicle in the intersection). In this case, the optimal signal switch time is equal to the minimum green time.

For each simulation of ACO algorithms, 100 trials are used. In each trial, every edge of the paths starts with equal pheromone level. During one iteration of the trial, every ant constructs a solution and consistently updates the pheromone. The pheromone level on the optimal path is plotted after each iteration; and the average performance of 100 trials is also shown as illustrated in Figure 6-1 to Figure 6-8.

An important issue in ACO algorithm is the number of ants used. Too little or too large number of ants do not lead to accurate outcomes either. One ant can be implemented, but it obviously does not take full advantage of ACO algorithm. On the other hand, a large number of ants increases the computational work per iteration. Moreover, the large amount of pheromone deposited does not allow significant bias toward to one path [7], and it is difficult to tell when the optimal solution is found because the pheromone level changes very slowly.

In the following simulations, the size of the local neighborhood is $T = 4$, which means that for local search, the solution search space is restricted to paths that are distance four or less from the best so far solution. The local search is performed at every 5th iteration. In equation (5.2), the heuristic weight function, the constant is $c = 5$. In equation (4.1), α and β are both equal to 1. In the Rank-based Ant System (RBAS), using equation (4.8), the elitist weight is $e = 10$ and the top 10 ants, $w = 10$, are used to update. The pheromone evaporation coefficients of $\rho=0.2$ and $\rho=0.4$ are compared. The number of iteration is 100, and the different numbers of ants, 5, 10, and 25 are used respectively. The following Table 6-2 shows the ACO parameters applied in simulations.

Parameter	Value
Number of Trial (for convergence)	100
Number of Iteration (for convergence)	100
Local Neighborhood (T)	4
Local Step	5
α	1
β	1
c	5
e (elitist weight)	10
w (rank weight)	10
ρ (evaporation coefficients)	0.2 / 0.4
Number of Ants	10 / 25 / 50

Table 6-2. ACO Parameters

6.2 Ant System (AS)

First of all, the original ant system algorithm is analyzed with 10, 25, and 50 ants individually. As seen in Figures 6-1 to 6-4, the pheromone levels tend to concentrate faster when only 10 ants are used. This becomes more apparent if $\rho=0.4$ (Figures 6-5 to 6-8). That is when more pheromone is evaporated on each iteration, pheromone levels converge faster. Moreover, the results of most trials do not converge to the optimal solution and this is not accurate enough.

Figure 6-1. Ant System rate of convergence with 10 ants, $\rho=0.2$

Figure 6-2. Ant System rate of convergence with 25 ants, $\rho=0.2$

Figure 6-3. Ant System rate of convergence with 50 ants, $\rho=0.2$

Figure 6-4. Ant System average rate of convergence with $\rho=0.2$

Figure 6-5. Ant System rate of convergence with 10 ants, $\rho=0.4$

Figure 6-6. Ant System rate of convergence with 25 ants, $\rho=0.4$

Figure 6-7. Ant System rate of convergence with 50 ants, $\rho=0.4$

Figure 6-8. Ant System average rate of convergence with $\rho=0.4$

6.3 Rank-based Ant System (RBAS)

The Rank-based Ant System algorithm is examined in this section. When $\rho=0.2$, the algorithm converges faster with 50 ants as seen in Figures 6-9 to 6-12. When $\rho=0.4$, Figures 6-13 to 6-16 indicate the convergence is not consistent, especially when more ants are used.

Figure 6-9. Rank-based Ant System rate of convergence with 10 ants, $\rho=0.2$

Figure 6-10. Rank-based Ant System rate of convergence with 25 ants, $\rho=0.2$

Figure 6-11. Rank-based Ant System rate of convergence with 50 ants, $\rho=0.2$

Figure 6-12. Rank-based Ant System rate of convergence with $\rho=0.2$

Figure 6-13. Rank-based Ant System rate of convergence with 10 ants, $\rho=0.4$

Figure 6-14. Rank-based Ant System rate of convergence with 25 ants, $\rho=0.4$

Figure 6-15. Rank-based Ant System rate of convergence with 50 ants, $\rho=0.4$

Figure 6-16. Rank-based Ant System rate of convergence with $\rho=0.4$

6.4 The Average Delay

In this section, the simulation results on the comparison of the ACO algorithm with the traditional Fully Actuated Control (FAC) are presented. The traffic simulation for each algorithm runs for twenty minutes first in order to reach a steady state and reduce the effect of initial conditions. Then, the delay time of each vehicle is recorded for the next twenty minutes. A vehicle's delay is defined to be its departure time minus its arrival time. At the end of each simulation, the average delay over the second twenty minutes is recorded. The average delay is:

$$\text{Average Delay} = \frac{\sum_i (dt_i - at_i)}{N} \quad (6.1)$$

where at_i and dt_i is the arrival time and departure time of the i^{th} vehicle respectively, and N is the total number of arrival vehicles during this interval. All delays are recorded in seconds.

Both Ant Colony Optimization (ACO) algorithm and Fully Actuated Control (FAC) are run on fifty sets of vehicle arrival data with two different left turn ration. The resolution of the simulation is 0.01 seconds.

As shown in Figure 6-17, the Fully Actuated control performs better when traffic volume is low and the left turn ration is 1. When the traffic flows is 3200 vehicles per hour, the ACO algorithm reduces the average delay time better. Figure 6-18 plots the upper and lower bounds of the average delay with left turn ration is 1. Figure 6-19 has the average queue length with left turn ratio is 1, and Figure 6-20 plots the queue length upper and lower bounds with left turn ratio is 1. In Figure 6-21, the ACO has better performance with left turn ratio is 0.5 when traffic flows reaches 3000 vehicles per hour. Figure 6-22 also plots the upper and lower bounds of left turn

ration which is 0.5. Figure 6-23 shows the average queue length with left turn ratio is 0.5. Figure 6-24 plots the queue length upper and lower bounds with left turn ratio is 0.5. Table 6-3 to Table 6-10 represent the mean (average), maximum, minimum, standard deviation, and variance for average delay and queue length with both left turn ratio is 1 and 0.5 (All details for each trial are given in Appendix A).

The ACO algorithm simulations run faster than real-time; therefore, it can be implemented in real time system. It takes fifteen minutes to run twenty minutes of simulation of the rank-based ant system with local search, elitist ant, and heuristic weights.

Figure 6-17. Average delay for (LT/T)=1

Figure 6-18. Upper and lower bounds on average delay for (LT/T)=1

Figure 6-19. Average queue length for (LT/T)=1

Figure 6-20. Upper and lower bounds on queue length for (LT/T)=1

Figure 6-21. Average delay for (LT/T)=0.5

Figure 6-22. Upper and lower bounds on average delay for $(LT/T)=0.5$

Figure 6-23. Average queue length for $(LT/T)=0.5$

Figure 6-24. Upper and lower bounds on queue length for $(L/T/T)=0.5$

volume	1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
MEAN	10.29	10.91	11.74	12.38	14.20	16.40	17.52	22.89	26.01	46.67	59.40
MAX	11.05	12.01	13.75	14.58	16.21	19.01	20.67	28.15	34.91	58.05	76.68
MIN	8.50	10.08	10.29	10.38	11.57	14.30	14.60	17.37	17.59	37.04	43.46
SD	0.55	0.42	0.87	0.96	1.05	1.23	1.26	2.56	3.98	5.70	6.42
VAR	0.31	0.18	0.76	0.93	1.10	1.51	1.59	6.53	15.85	32.47	41.28

Table 6-3. Delay time (ACO with LT/T=1)

volume	1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
MEAN	9.53	9.85	10.28	10.91	11.71	13.13	15.58	22.05	33.95	50.87	67.90
MAX	10.88	10.70	11.13	12.01	12.53	15.56	20.74	30.46	51.96	75.80	91.97
MIN	8.72	8.79	9.40	9.92	10.90	11.30	13.10	14.47	25.98	39.01	53.79
SD	0.44	0.40	0.33	0.45	0.41	0.90	1.52	3.55	5.16	6.24	8.86
VAR	0.2	0.16	0.11	0.20	0.17	0.81	2.30	12.58	26.63	38.90	78.58

Table 6-4. Delay time (FAC with LT/T=1)

volume	1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
MEAN	1.30	1.55	1.73	1.95	2.24	2.66	2.91	3.75	4.49	6.98	8.39
MAX	1.42	1.65	1.88	2.07	2.54	2.93	3.31	4.32	6.05	8.99	11.76
MIN	1.13	1.43	1.58	1.81	1.99	2.33	2.53	3.08	3.09	4.52	5.45
SD	0.07	0.05	0.07	0.06	0.13	0.14	0.18	0.26	0.67	0.98	1.16
VAR	0.00	0.00	0.01	0.00	0.02	0.02	0.03	0.07	0.45	0.96	1.34

Table 6-5. Queue length (ACO with LT/T=1)

Volume	1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
MEAN	1.47	1.61	1.77	1.97	2.20	2.54	3.07	4.41	7.43	11.23	15.37
MAX	1.54	1.67	1.88	2.09	2.34	2.96	3.63	6.91	9.17	14.90	20.53
MIN	1.40	1.51	1.68	1.88	2.07	2.34	2.79	3.32	5.23	9.55	12.10
SD	0.04	0.03	0.04	0.06	0.06	0.11	0.19	0.60	0.80	1.04	1.87
VAR	0.00	0.00	0.00	0.00	0.00	0.01	0.03	0.35	0.89	1.10	3.50

Table 6-6. Queue length (FAC with LT/T=1)

volume	1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
MEAN	10.06	10.79	11.56	12.57	14.19	15.36	18.08	22.90	36.86	47.54	62.08
MAX	11.00	11.94	12.58	13.99	16.20	17.61	22.12	29.34	47.76	58.94	78.25
MIN	9.05	9.60	10.46	11.22	11.55	12.63	13.65	17.49	23.44	33.18	45.90
SD	0.47	0.50	0.53	0.65	0.95	1.22	2.23	2.69	5.84	6.10	6.84
VAR	0.22	0.25	0.28	0.42	0.90	1.50	4.98	7.23	34.08	37.16	46.72

Table 6-7. Delay time (ACO with LT/T=0.5)

volume	1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
MEAN	9.07	9.49	10.14	10.63	11.92	13.79	18.16	29.12	40.07	57.49	72.58
MAX	9.80	10.22	11.49	11.74	13.36	17.30	26.09	42.66	54.15	70.90	92.89
MIN	8.23	8.75	9.25	9.90	10.79	11.17	14.00	19.01	25.70	45.32	57.38
SD	0.39	0.34	0.46	0.44	0.68	1.45	2.85	5.59	5.78	6.15	8.21
VAR	0.15	0.12	0.21	0.19	0.46	2.10	8.15	31.25	33.38	37.79	67.41

Table 6-8. Delay time (FAC with LT/T=0.5)

volume	1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
MEAN	1.27	1.47	1.70	1.93	2.23	2.43	2.96	3.61	5.59	7.19	9.31
MAX	1.40	1.55	1.86	2.19	2.45	2.78	3.49	4.69	7.11	9.56	11.79
MIN	1.15	1.36	1.49	1.71	2.01	2.09	2.33	2.64	4.15	4.52	6.76
SD	0.06	0.04	0.08	0.12	0.10	0.17	0.31	0.49	0.73	0.96	1.06
VAR	0.00	0.00	0.01	0.01	0.01	0.03	0.10	0.24	0.54	0.93	1.12

Table 6-9. Queue length (ACO with $LT/T=0.5$)

volume	1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
MEAN	1.39	1.54	1.71	1.91	2.20	2.59	3.46	5.69	7.41	11.80	16.18
MAX	1.47	1.61	1.88	2.03	2.38	3.08	4.22	8.66	10.22	13.87	18.88
MIN	1.33	1.42	1.58	1.81	2.03	2.18	2.86	4.23	5.20	9.54	13.89
SD	0.04	0.03	0.08	0.05	0.08	0.22	0.33	0.83	1.12	0.85	1.27
VAR	0.00	0.00	0.01	0.00	0.01	0.05	0.11	0.69	1.24	0.72	1.61

Table 6-10. Queue length (FAC with $LT/T=0.5$)

Chapter 7 Concluding Remarks and Future Research

To reduce vehicle delay, fuel consumption, and avoid the traffic congestion, traffic signal control is an effective way to regulate traffic follows. However, traffic signal optimization is complicated due to the nonlinear and stochastic native of traffic flows. In this thesis, ACO algorithm is applied to solve this problem.

A single isolated intersection is considered in this research, and the next step is to apply ACO algorithm to a five intersections network with a central intersection and four surrounding intersections. Unlike the case of an isolated intersection, the interactions between intersections must be included in the traffic model for a network.

Figure 7-1. A typical elementary traffic network with five intersections

As shown in Chapter 6, ACO algorithm with Rank-based ant system performs well when traffic volume is high, and by experimenting different constants of the algorithm should be able

to improve the performance. The other direction of future work is to combine ACO with other traditional algorithms for best performance with different traffic demand.

References

- [1] Ahuja, S., Bell, M. G. H. and Schmöcker, J.-D., "Multi-objective signal control of urban junctions - Framework and a London case study," *Transportation Research Part C: Emerging Technologies*, vol. 16, pp. 454-470, 2008.
- [2] Almeida, F. and Ramos, V., "Artificial ant colonies in digital image habitats - a mass behavior effect Study on pattern recognition. " Proceedings of ANTS'2000 - 2nd International Workshop on Ant Algorithms, Brussels, Belgium, 2000
- [3] Angus, D., "Ant colony Optimization: From Biological inspiration to an Algorithmic Framework," *Technical Report, CISCP*, 2006.
- [4] Bruten, J., Holland, O., Rothkrantz, L. and Schoonderwoerd, R., "Ant-based load balancing in telecommunications networks. " *Adaptive Behavior*, 5(2), 1997.
- [5] Bullnheimer, B., Hartl, R. F., and Strauss, C., "An improved Ant System algorithm for the Vehicle Routing Problem," *Annals of Operations Research*, vol. 89, pp. 319-328, 1999.
- [6] Dorigo, M., "Optimization, Learning and Natural Algorithms, " PhD thesis, Politecnico di Milano, Italie, 1992.
- [7] Dorigo, M. and Gambardella, L. M., "Ant colonies for the traveling salesman problem," *Biosystems*, vol. 43, pp. 73-81, 1997.
- [8] Dorigo, M. and Stutzle, T., "ACO algorithm for the quadratic assignment problem," *New Idea In Optimization*, McGraw-Hill, 1999.
- [9] Dorigo, M., Stutzle, T., *Ant colony optimization*. Cambridge, Massachusetts: MIT Press, 2004.
- [10] Eisele, B., Lomax, T., and Schrank, D., " *TTI's 2011 CONGESTED CORRIDORS REPORT Powered by INRIX Traffic Data*," Texas Transportation Institute, The Texas A&M University System November 2011
- [11] Goss, S. Aron, J.-L. Deneubourg et J.-M. Pasteels, "Self-organized shortcuts in the Argentine ant, *Naturwissenschaften*, " volume 76, pages 579-581, 1989.
- [12] Hunt, P. B., Robertson, D. I., et al., "The SCOOT on-line traffic signal optimization technique," *Traffic engineering and control*, April 1982.
- [13] Kuntz , P. and Snyers, D., "Emergent colonization and graph partitioning. Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3", MIT Press: Cambridge, MA. , 1994

- [14] Lin, B.M.T., Shyu, S.J., and Yin, P.Y., "An ant colony optimization algorithm for the minimum weight vertex cover problem. " to appear in *Annals of Operations Research*, 2003
- [15] Lin, B.M.T., Shyu, S.J., and Yin, P.Y., "Ant-Tree: an ant colony optimization approach to the generalized minimum spanning tree problem. " *Journal of Experimental and Theoretical Artificial Intelligence*, 15, 2003
- [16] Lowrie, P., "The Sydney coordinated adaptive control systems - principles, methodology, algorithms," *IEE conference publication*, vol. 207, 1982.
- [17] Mbede, J. B., Song, J., Wei, W., Zhang,Y. and Zhang, Z., "Traffic signal control using fuzzy logic and MOGA," *Proceedings of the IEEE conference on systems, man, and cybernetics*, 2001.
- [18] Ramanathan, B. V., "Actuated signal network simulation program," Institute of Transportation Studies University of California Irvine 1993.
- [19] Renfrew, David, "Traffic Signal Control With Ant Colony Optimization," California Polytechnic University, San Luis Obispo, 2009.
- [20] Renfrew, David, Yu, Xiao-Hua, Member, IEEE, "Traffic Signal Control with Swarm Intelligence," Department of Electrical Engineering California Polytechnic State University, San Luis Obispo, 2008.
- [21] Ross, S. M., *Applied Probability Models with Optimization Applications*. San Francisco: Holden-Day, 1970.
- [22] Stubberud, Allen R., Yu, Xiao-Hua, " A New Control Strategy for A Signalized Traffic Intersection," Department of Electrical Engineering and Computer Engineering University of California, Irvine, 1998.
- [23] Transportation research laboratory, "Traffic network study tool: TRANSYT-7F software summary," University of Florida, 1987.
- [24] Tsubota, Ogawa Y., H. Kawashima, "A Study on the Introduction of Dual-ring Traffic Signal Control for Mitigating Traffic Impacts," Division of Natural Sciences, J. F. Oberlin University, Tokyo.
- [25] Yates, R. D., Goodman, D.J., *Probability and Stochastic Processes*, second ed. New York: Wiley, 2005.
- [26] Yu, X.-H., "Decentralized adaptive signal control for traffic networks," in *Transportation research trends*, P. O. Inweldi, Ed. New York: Nova Science Publishers, 2008.

- [27] Yu, X.-H., and W. W. Recker, "Stochastic adaptive control model for traffic signal systems," *Transportation Research Part C: Emerging Technologies*, vol. 14, pp. 263-282, 2006.

Appendix A Tables of Delay Time and Waiting Queues

The following tables show the delay time and waiting queues in ACO and FAC. The first row gives the total intersection vehicles per hour, and the rest of columns give the delay time and waiting queues during each trial.

Table A-1 ACO delay time with LT/T=1

1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
10.39	10.09	11.09	12.11	13.13	15.13	16.65	23.18	21.48	45.09	56.76
10.91	10.75	11.88	11.21	15.13	17.70	20.67	24.53	22.90	48.83	66.18
10.64	11.38	12.11	12.61	14.99	17.65	17.09	22.06	26.52	45.56	61.55
10.49	11.13	10.66	13.56	12.55	17.72	17.94	24.09	26.39	52.63	54.13
9.82	10.63	11.97	13.49	13.46	16.06	19.09	18.75	26.28	38.21	63.24
10.18	10.48	12.62	13.98	14.45	17.33	16.87	25.19	21.09	45.43	70.64
9.54	11.11	12.51	10.38	12.96	16.59	19.58	24.87	28.76	51.30	68.76
10.90	10.68	11.75	12.17	15.69	14.30	16.02	25.08	27.40	48.30	60.62
11.05	10.08	10.47	13.40	14.14	17.45	16.18	22.67	24.31	53.54	66.86
9.29	11.50	13.12	11.39	12.91	17.42	17.79	22.48	18.99	41.40	65.02
10.38	11.70	11.92	11.76	12.58	15.36	17.03	27.08	28.12	38.15	53.86
9.67	10.75	10.54	10.83	14.07	14.89	18.21	26.18	29.48	46.23	47.88
10.63	10.33	11.52	10.69	13.49	17.82	18.75	22.06	22.40	41.01	56.51
10.39	10.72	11.99	11.49	15.30	16.29	17.19	24.96	23.38	56.79	52.10
10.72	10.87	10.99	11.68	13.66	14.82	18.39	22.77	21.44	42.25	65.27
9.90	10.76	11.30	10.44	12.44	16.43	18.85	26.79	28.31	39.69	59.31
10.01	10.27	11.52	11.87	15.13	16.50	17.20	21.33	28.56	52.92	62.76
10.27	11.03	11.70	12.02	14.77	17.35	17.90	25.14	25.61	47.27	56.72
8.50	10.47	12.25	12.51	16.21	17.83	19.29	17.98	30.66	52.84	54.80
10.88	11.19	11.07	12.05	14.03	16.91	17.87	19.74	23.06	49.26	67.46
11.04	11.13	11.46	12.09	14.31	15.22	15.51	20.36	25.84	46.74	43.46
10.92	10.57	11.87	12.05	14.31	18.10	18.04	21.34	27.45	41.87	62.88
9.52	11.40	13.10	13.63	14.40	15.12	14.60	22.68	17.59	52.70	56.46
9.76	11.32	12.10	12.31	13.46	16.92	16.31	27.34	27.47	37.04	54.08

9.42	11.01	12.85	12.22	15.18	15.69	18.42	24.39	30.67	40.04	68.39
10.52	10.84	11.44	11.92	13.55	16.65	18.09	22.56	22.84	43.07	60.96
9.96	10.80	13.75	14.20	14.81	18.87	18.47	22.34	18.41	37.55	51.08
10.62	11.48	12.68	13.19	14.92	14.34	18.70	22.62	32.47	41.81	59.13
10.53	11.00	10.40	14.58	13.44	17.19	17.71	17.37	25.04	46.68	60.30
10.90	10.96	10.32	11.79	13.39	15.69	17.77	19.93	27.69	41.77	58.16
10.71	10.43	13.17	13.92	16.00	14.71	15.42	19.49	22.40	45.33	60.69
10.91	12.01	11.07	10.91	14.98	18.14	16.15	24.75	25.93	51.89	55.99
10.08	11.24	11.05	12.89	14.11	14.91	17.27	21.87	33.62	47.06	62.64
10.00	10.75	11.64	13.22	15.05	15.22	18.26	21.32	25.52	49.22	50.01
10.31	10.97	10.85	14.16	15.00	15.88	17.08	25.14	34.63	49.69	53.84
10.02	10.48	10.66	12.31	15.79	17.34	15.70	27.90	28.53	40.90	66.02
9.64	11.23	11.93	11.91	13.02	16.58	18.14	18.89	21.45	47.11	76.68
10.80	10.89	13.41	13.01	13.21	15.88	17.62	21.98	27.00	40.25	54.31
10.27	11.36	10.84	12.04	14.92	14.89	17.57	23.23	28.08	53.08	55.18
10.54	11.28	11.68	13.09	15.67	17.57	16.88	22.15	30.66	48.60	57.66
10.86	10.50	11.98	12.26	13.92	17.55	17.31	24.88	26.53	53.07	61.02
9.82	10.43	12.83	12.18	12.94	17.57	16.79	20.14	24.38	49.22	70.19
10.91	10.76	11.86	12.57	14.49	16.27	18.62	23.39	27.43	37.66	50.85
10.06	10.90	12.09	12.34	15.00	19.01	18.62	24.20	30.73	54.08	52.92
9.44	11.44	11.70	12.28	11.57	14.80	18.67	28.15	24.55	40.30	60.96
10.20	10.84	12.72	12.65	15.90	16.53	16.80	24.54	26.03	58.05	61.04
10.35	11.45	12.32	12.13	13.96	15.09	14.75	21.96	23.49	53.86	55.20
9.95	10.97	10.37	12.29	13.69	15.87	15.88	20.35	34.91	54.52	59.66
10.77	10.31	10.29	12.17	14.05	15.41	17.14	21.23	20.87	42.10	62.07
10.90	10.77	11.60	12.93	14.01	15.62	19.20	21.19	22.90	51.49	57.97

Table A-2 FAC delay time with LT/T=1

1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
9.99	9.31	10.01	10.81	11.12	11.99	16.18	14.47	34.54	52.42	72.70
9.29	9.88	10.27	10.79	12.17	15.56	14.95	22.15	35.47	46.09	59.91
9.32	9.61	10.12	10.93	11.16	14.48	15.77	18.98	34.13	53.66	79.47
9.81	9.86	9.81	11.40	11.95	12.48	14.48	23.51	34.04	55.71	69.47

9.04	9.77	10.35	11.14	12.36	12.56	19.06	20.67	30.90	45.93	67.54
10.13	8.94	10.05	10.47	10.90	11.91	15.01	18.72	51.96	45.60	78.97
9.87	9.64	10.69	11.11	12.07	12.67	15.45	18.41	33.29	47.66	72.81
10.00	9.55	9.40	9.92	11.46	12.70	18.94	24.24	34.94	44.67	71.84
9.66	9.80	10.43	12.01	12.24	13.46	14.94	26.21	32.89	55.24	64.44
9.19	9.70	10.46	10.50	12.07	12.57	14.79	19.39	33.31	51.73	54.58
8.89	9.88	10.66	11.54	11.98	14.79	16.45	19.35	30.49	55.19	62.50
9.26	9.95	10.48	10.46	11.75	14.49	14.03	23.79	36.71	51.36	61.28
10.21	10.51	10.11	10.95	11.54	13.34	14.80	24.13	34.60	55.97	84.03
8.94	10.46	10.26	10.93	12.01	14.17	14.34	16.69	36.72	47.95	74.44
9.44	9.74	10.56	11.49	11.99	13.06	15.46	16.73	45.98	58.76	65.72
10.12	10.19	9.74	10.90	11.78	13.68	13.99	25.39	27.90	45.07	60.12
9.72	10.07	10.26	11.28	12.00	13.37	14.98	17.76	34.67	53.91	71.67
9.33	9.75	9.86	11.53	11.06	12.51	16.82	29.11	33.86	39.01	91.97
9.24	9.45	10.45	11.54	11.02	12.43	15.91	25.25	38.73	61.84	76.85
9.50	10.17	10.40	10.89	11.73	12.91	14.28	22.63	31.31	40.60	61.83
9.57	10.01	11.13	10.41	11.20	11.75	14.73	23.06	37.34	49.48	63.22
9.60	10.16	10.14	11.10	11.44	12.58	15.36	17.71	33.15	60.28	53.94
10.62	9.40	10.71	11.58	12.13	11.30	14.59	19.84	29.55	48.67	79.82
9.23	9.51	10.58	10.85	11.83	12.69	17.59	25.68	34.23	48.22	64.86
9.86	9.45	10.55	11.27	12.00	12.55	18.04	25.55	38.62	55.57	65.28
9.44	10.19	9.90	10.45	12.04	13.27	17.20	30.46	28.81	48.22	58.67
9.42	9.68	9.96	11.66	11.98	14.74	15.30	22.28	27.10	49.69	65.52
10.17	8.79	10.96	10.63	10.96	13.28	14.77	19.83	29.59	59.25	70.35
9.14	9.42	9.92	10.11	11.44	12.47	15.24	19.61	40.44	51.94	83.91
9.26	10.05	10.46	10.86	11.85	12.79	15.66	20.63	37.91	45.36	80.63
9.08	10.20	10.29	11.89	11.93	13.08	15.80	18.31	26.08	40.76	69.46
9.58	9.82	10.00	10.45	12.38	13.15	20.74	19.64	25.98	43.41	63.99
9.21	9.54	10.35	10.97	11.47	12.53	16.35	28.87	26.98	45.35	69.76
9.09	9.21	10.77	10.25	11.24	13.79	13.40	20.93	31.43	48.22	59.47
9.82	10.47	10.43	10.94	12.02	13.85	15.31	19.81	30.93	47.03	75.66
9.73	10.36	10.24	10.65	12.02	12.24	13.94	25.72	29.04	52.68	66.04
9.25	10.60	10.15	10.99	11.29	12.53	14.48	21.10	26.93	75.98	64.21
10.88	9.91	9.81	11.12	11.50	12.08	13.92	22.03	30.66	46.75	79.02
9.79	10.70	10.33	10.78	12.03	12.60	15.49	24.96	40.03	51.34	56.06
9.45	9.80	10.35	10.32	11.52	13.60	14.92	30.28	34.21	49.48	77.77
9.10	9.95	10.34	10.43	11.59	12.94	13.10	22.29	37.07	52.73	53.80
8.72	9.66	10.07	10.71	11.32	13.34	16.53	24.35	40.51	56.40	57.95

9.58	9.60	10.18	10.77	12.04	13.70	17.16	22.33	41.21	52.32	56.37
9.09	10.25	10.18	10.69	11.21	14.02	14.98	18.64	32.49	54.58	73.26
9.41	9.77	10.12	10.44	11.68	12.18	17.14	23.00	37.17	53.41	66.31
9.86	9.67	10.30	10.55	11.14	12.58	14.76	23.32	37.43	46.53	53.79
8.97	9.77	10.22	10.99	12.53	14.39	13.38	22.15	31.17	50.10	64.77
9.46	10.31	10.64	10.89	11.88	13.63	16.87	19.63	37.58	47.33	69.07
9.26	9.97	9.80	11.27	11.81	13.01	15.41	23.41	27.10	54.74	67.78
9.91	9.79	10.50	10.87	11.58	14.85	16.18	19.62	30.18	49.09	62.35

Table A-3 ACO waiting queues with LT/T=1

1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
1.31	1.44	1.67	1.93	2.11	2.54	2.74	3.79	3.78	6.69	8.31
1.40	1.52	1.75	1.87	2.33	2.82	3.31	3.89	3.89	7.26	9.29
1.35	1.61	1.77	1.99	2.31	2.81	2.83	3.66	4.59	6.86	8.80
1.32	1.57	1.66	2.02	2.03	2.83	2.95	3.88	4.61	7.82	7.39
1.25	1.51	1.75	2.02	2.15	2.63	3.14	3.34	4.57	5.86	8.97
1.28	1.50	1.79	2.03	2.26	2.71	2.78	4.01	3.67	6.85	10.84
1.22	1.57	1.78	1.81	2.09	2.69	3.27	3.89	5.05	7.53	9.91
1.37	1.52	1.74	1.94	2.44	2.33	2.70	3.97	4.68	7.11	8.61
1.42	1.43	1.64	2.02	2.24	2.78	2.72	3.73	4.09	8.09	9.40
1.17	1.65	1.84	1.88	2.09	2.77	2.93	3.69	3.46	6.30	9.01
1.30	1.65	1.75	1.89	2.08	2.56	2.79	4.11	4.93	5.25	7.35
1.23	1.52	1.64	1.86	2.22	2.52	3.02	4.05	5.10	6.90	5.93
1.35	1.47	1.72	1.86	2.16	2.83	3.11	3.67	3.78	6.19	8.29
1.31	1.52	1.76	1.88	2.43	2.65	2.86	3.96	4.04	8.59	7.14
1.35	1.55	1.67	1.88	2.18	2.50	3.05	3.77	3.77	6.69	9.08
1.25	1.53	1.69	1.83	2.00	2.66	3.13	4.05	4.92	6.07	8.46
1.27	1.47	1.71	1.90	2.35	2.65	2.88	3.62	5.02	8.05	8.97
1.29	1.57	1.74	1.91	2.28	2.76	2.95	3.98	4.44	7.09	8.32
1.13	1.50	1.76	1.99	2.54	2.85	3.23	3.16	5.12	7.88	7.53
1.36	1.57	1.67	1.92	2.22	2.71	2.95	3.39	4.00	7.44	9.57
1.42	1.57	1.71	1.93	2.26	2.55	2.65	3.55	4.45	6.97	5.45
1.41	1.51	1.74	1.91	2.24	2.85	2.96	3.66	4.81	6.40	8.97

1.20	1.62	1.83	2.03	2.25	2.52	2.53	3.75	3.09	7.77	8.13
1.24	1.60	1.76	1.96	2.15	2.71	2.73	4.17	4.84	4.52	7.38
1.18	1.56	1.83	1.95	2.35	2.59	3.04	3.88	5.30	6.11	9.77
1.31	1.54	1.68	1.90	2.18	2.71	2.99	3.71	3.86	6.66	8.68
1.26	1.54	1.88	2.05	2.28	2.88	3.06	3.68	3.23	4.83	7.08
1.34	1.64	1.82	2.01	2.29	2.44	3.09	3.70	5.49	6.40	8.39
1.32	1.57	1.63	2.07	2.15	2.72	2.93	3.08	4.22	6.93	8.57
1.37	1.56	1.60	1.89	2.12	2.59	2.95	3.43	4.88	6.35	8.39
1.36	1.49	1.87	2.03	2.51	2.45	2.61	3.38	3.78	6.70	8.57
1.37	1.65	1.67	1.86	2.29	2.86	2.71	3.89	4.46	7.74	8.06
1.28	1.57	1.67	2.00	2.24	2.52	2.90	3.65	5.53	7.05	8.82
1.26	1.52	1.73	2.01	2.32	2.55	3.03	3.62	4.33	7.43	6.43
1.29	1.56	1.67	2.03	2.31	2.62	2.80	3.99	5.94	7.53	7.18
1.26	1.50	1.65	1.97	2.45	2.74	2.65	4.28	4.97	6.18	9.25
1.22	1.58	1.75	1.91	2.10	2.68	3.00	3.37	3.76	7.04	11.76
1.36	1.56	1.88	2.00	2.11	2.62	2.90	3.66	4.62	6.13	7.52
1.28	1.61	1.67	1.91	2.28	2.52	2.90	3.80	4.88	8.08	7.66
1.33	1.60	1.73	2.01	2.43	2.81	2.79	3.67	5.17	7.18	8.33
1.36	1.51	1.75	1.96	2.18	2.79	2.89	3.92	4.61	8.07	8.69
1.25	1.49	1.82	1.95	2.09	2.78	2.77	3.53	4.21	7.35	10.31
1.38	1.54	1.75	1.99	2.26	2.63	3.07	3.82	4.78	4.85	6.71
1.28	1.56	1.76	1.96	2.30	2.93	3.06	3.86	5.30	8.26	7.17
1.20	1.63	1.74	1.96	1.99	2.45	3.08	4.32	4.21	6.17	8.65
1.28	1.55	1.82	1.99	2.50	2.67	2.77	3.89	4.48	8.99	8.73
1.29	1.63	1.77	1.93	2.20	2.52	2.60	3.66	4.07	8.09	7.69
1.26	1.56	1.62	1.96	2.19	2.61	2.66	3.55	6.05	8.51	8.54
1.36	1.47	1.58	1.93	2.22	2.57	2.83	3.61	3.61	6.60	8.87
1.37	1.53	1.72	2.01	2.20	2.58	3.23	3.61	3.90	7.68	8.36

Table A-4 FAC waiting queues with LT/T=1

1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
1.50	1.61	1.80	1.89	2.18	2.46	3.10	3.32	8.08	10.73	16.11
1.46	1.60	1.80	1.99	2.26	2.81	3.07	4.33	8.04	10.26	14.66

1.44	1.61	1.75	1.98	2.17	2.63	3.29	4.53	6.78	11.65	16.27
1.47	1.63	1.78	2.00	2.21	2.55	2.94	4.92	7.91	11.81	15.56
1.47	1.60	1.77	2.07	2.09	2.47	3.53	4.48	7.38	10.34	14.14
1.54	1.57	1.81	1.93	2.10	2.36	3.28	3.96	8.95	11.28	20.28
1.51	1.58	1.83	2.03	2.26	2.49	2.87	4.27	7.69	9.93	15.74
1.54	1.63	1.71	1.88	2.24	2.44	3.32	4.52	7.12	9.98	16.40
1.46	1.55	1.78	2.03	2.20	2.55	3.00	5.60	7.13	11.79	16.23
1.51	1.57	1.88	1.92	2.16	2.50	3.20	3.61	7.66	10.98	12.10
1.44	1.60	1.85	2.01	2.23	2.76	3.06	4.13	7.35	11.30	13.31
1.45	1.60	1.72	1.90	2.11	2.96	3.12	4.40	7.20	10.67	14.21
1.50	1.66	1.74	1.90	2.28	2.49	2.94	4.60	6.73	11.96	20.53
1.43	1.65	1.76	1.92	2.27	2.52	2.88	4.01	7.56	11.88	16.88
1.42	1.58	1.81	2.03	2.22	2.58	3.20	3.57	7.74	12.92	14.78
1.45	1.63	1.73	1.91	2.19	2.58	2.83	4.60	6.39	10.55	14.04
1.48	1.62	1.70	2.08	2.28	2.52	3.04	3.95	8.56	12.33	15.94
1.49	1.59	1.75	2.03	2.07	2.56	3.04	5.69	7.21	9.55	17.57
1.47	1.57	1.81	1.96	2.12	2.47	3.07	4.64	9.05	12.65	16.38
1.42	1.65	1.77	1.96	2.19	2.52	2.94	4.70	6.90	10.65	13.65
1.49	1.61	1.80	1.89	2.19	2.34	3.05	4.35	8.35	10.16	15.53
1.46	1.64	1.71	2.03	2.09	2.44	2.91	4.15	6.02	11.90	13.30
1.52	1.64	1.82	2.09	2.24	2.36	2.84	4.53	7.09	10.88	19.14
1.48	1.59	1.77	1.94	2.28	2.53	2.96	4.75	7.78	10.78	15.06
1.53	1.60	1.77	2.04	2.24	2.49	3.32	5.17	8.77	11.84	14.36
1.44	1.65	1.68	1.91	2.26	2.64	3.18	6.91	5.72	10.30	13.65
1.51	1.59	1.78	2.04	2.23	2.59	3.45	4.18	5.23	10.84	14.26
1.51	1.51	1.83	1.96	2.09	2.50	3.02	3.98	6.73	12.63	16.52
1.43	1.56	1.73	1.89	2.24	2.57	3.19	4.57	9.17	11.93	18.35
1.45	1.66	1.77	2.01	2.21	2.54	3.30	4.04	7.35	9.96	15.41
1.44	1.62	1.79	2.01	2.19	2.53	2.91	4.62	6.57	10.77	14.90
1.44	1.61	1.77	1.95	2.29	2.61	3.63	4.30	6.69	9.59	15.36
1.44	1.56	1.80	1.94	2.25	2.49	3.15	5.39	6.59	10.01	16.47
1.40	1.59	1.85	1.88	2.16	2.65	2.87	3.83	7.17	10.61	13.18
1.52	1.59	1.74	1.93	2.31	2.58	2.98	3.69	7.55	11.07	17.38
1.52	1.62	1.76	1.98	2.23	2.38	2.96	4.50	6.50	11.39	14.19
1.45	1.61	1.77	2.05	2.22	2.38	2.81	3.89	6.76	14.90	14.06
1.50	1.63	1.69	1.97	2.29	2.50	2.86	4.22	6.85	11.60	18.00
1.52	1.66	1.76	1.99	2.22	2.50	3.12	4.35	8.71	10.20	13.44
1.50	1.54	1.78	1.97	2.13	2.52	2.94	4.95	8.46	10.02	17.81

1.43	1.64	1.76	1.98	2.16	2.58	2.97	4.05	8.27	12.58	12.46
1.41	1.64	1.78	2.00	2.18	2.70	3.04	4.65	8.80	12.76	13.30
1.42	1.59	1.76	1.90	2.17	2.62	3.18	4.19	7.53	12.79	14.70
1.50	1.64	1.73	1.88	2.23	2.74	2.79	4.17	6.77	11.43	15.97
1.50	1.59	1.78	1.99	2.07	2.52	3.19	4.38	7.89	11.38	14.52
1.52	1.61	1.75	2.02	2.17	2.44	2.90	4.23	8.51	11.05	13.33
1.45	1.61	1.78	1.96	2.34	2.43	2.96	3.93	6.39	10.63	15.53
1.40	1.67	1.85	1.99	2.21	2.60	3.11	4.23	8.13	10.81	15.01
1.46	1.62	1.69	2.03	2.18	2.55	2.96	4.28	7.28	12.05	14.73
1.47	1.57	1.77	1.99	2.14	2.58	3.04	4.08	6.63	11.52	13.74

Table A-5 ACO delay time with LT/T=0.5

1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
10.15	10.94	11.48	13.14	14.33	14.51	16.38	28.39	28.64	49.19	57.62
10.32	10.60	11.84	13.37	15.04	15.29	21.36	20.69	34.82	41.24	59.83
9.63	10.71	11.04	11.36	14.35	16.22	21.04	20.26	43.31	54.28	65.33
10.63	11.36	11.91	11.76	14.41	14.28	16.62	22.18	36.78	43.91	66.95
10.13	10.93	10.54	11.32	14.56	15.00	22.11	19.51	37.69	58.01	53.78
9.67	10.49	11.55	13.58	13.86	16.27	17.02	20.96	29.24	47.33	59.16
9.70	10.41	12.42	11.96	15.39	15.44	15.32	23.05	28.39	39.41	66.31
9.08	11.09	11.54	11.90	13.55	16.38	18.26	25.32	42.28	47.36	56.82
10.36	10.55	11.58	13.21	13.97	16.85	16.75	24.01	47.38	53.29	78.25
11.00	10.01	12.04	13.04	13.65	13.49	19.18	24.17	36.98	44.93	58.23
9.52	10.82	12.50	13.72	12.76	15.86	20.03	19.63	44.03	39.14	54.60
10.60	10.54	11.53	12.41	11.55	15.07	16.45	22.33	28.69	51.30	66.91
9.98	11.00	11.47	12.67	12.83	15.10	15.84	29.34	35.56	52.14	70.02
9.98	11.00	12.11	13.07	13.89	16.99	16.46	20.63	31.51	44.62	53.18
9.28	11.01	11.92	12.81	13.51	13.79	18.87	20.51	33.87	39.19	45.90
10.16	9.60	12.03	11.58	13.98	15.69	19.82	24.12	42.22	39.97	64.91
9.96	10.86	10.66	12.06	14.57	13.95	16.00	21.60	39.81	33.18	60.74
10.58	10.66	12.35	12.01	14.91	16.50	20.78	21.24	31.60	51.24	58.57
10.56	9.99	11.30	12.88	13.85	16.68	18.48	24.51	44.26	41.81	69.91
10.60	11.74	11.83	12.04	13.19	15.94	20.54	17.68	28.43	49.52	64.04

10.07	10.99	11.20	12.71	15.90	15.02	15.06	25.96	39.64	55.37	62.25
10.11	11.29	12.24	13.19	13.75	17.59	18.01	22.63	41.82	48.23	71.15
9.94	10.47	11.49	13.29	13.45	17.61	19.44	24.50	41.72	45.13	62.56
9.95	10.99	11.23	12.50	13.67	17.19	17.44	17.83	39.45	49.67	61.61
9.79	9.82	11.54	12.68	14.72	14.49	14.56	24.49	33.62	42.38	75.68
10.00	11.21	10.56	13.11	15.32	14.31	19.07	24.56	37.85	35.47	62.36
10.55	11.10	11.42	13.00	14.85	13.34	19.38	20.59	27.21	52.70	76.38
10.47	10.51	10.46	12.72	13.72	14.21	18.54	23.00	27.16	43.16	53.65
9.05	9.91	12.16	12.96	14.81	15.05	22.12	21.31	35.03	40.88	56.35
10.53	11.37	11.58	12.38	15.51	14.80	13.65	24.09	30.25	57.35	62.17
9.95	11.46	10.97	12.68	12.73	13.48	15.95	22.40	40.03	45.07	56.12
10.04	10.88	11.17	12.20	14.39	15.68	21.97	26.97	39.03	47.53	52.69
10.83	10.94	11.21	13.19	15.27	14.48	19.64	26.18	47.76	47.81	67.70
9.44	11.51	11.29	12.43	14.63	16.64	20.30	25.23	39.47	58.82	50.49
10.55	11.39	11.37	11.22	15.22	16.43	20.07	24.79	35.19	58.72	60.27
9.22	10.64	11.81	12.72	14.54	15.26	18.15	19.00	46.19	50.17	71.92
10.26	11.05	10.91	13.21	14.67	14.45	14.49	20.11	27.77	50.21	54.80
9.74	11.11	11.23	12.02	14.76	15.35	15.40	24.02	36.38	50.62	57.06
10.10	10.33	11.72	13.99	12.53	16.05	16.32	26.78	35.56	46.16	62.72
9.25	11.00	11.06	12.39	13.23	14.29	17.86	20.73	42.35	49.76	66.02
9.88	10.81	12.29	11.93	16.20	14.62	15.99	21.73	43.15	52.00	60.72
9.35	10.01	11.59	11.46	14.09	12.63	19.79	25.80	39.65	47.09	70.12
10.83	11.13	11.63	12.77	13.40	17.55	15.59	22.84	42.36	44.73	58.99
10.16	11.02	11.21	12.14	12.63	16.26	19.29	26.38	36.39	46.61	58.04
10.38	10.24	11.77	12.46	13.38	13.70	17.77	22.61	23.44	45.18	65.06
10.06	10.75	11.49	12.41	14.71	16.41	15.75	24.87	36.17	45.65	64.34
10.38	11.94	10.89	12.69	14.36	14.47	18.84	22.46	35.96	58.94	63.47
9.95	10.66	12.02	12.92	15.51	15.07	21.96	22.42	36.25	50.85	60.08
10.13	10.13	12.58	11.82	14.79	15.05	17.75	23.22	40.41	37.96	70.43
10.03	10.62	12.38	13.24	14.58	17.03	16.35	17.49	40.33	51.50	57.81

Table A-6 FAC delay time with LT/T=0.5

1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
------	------	------	------	------	------	------	------	------	------	------

9.79	9.26	10.35	11.49	11.77	11.39	19.64	29.60	32.15	62.21	81.26
9.73	9.13	9.73	10.26	12.98	16.08	16.74	23.25	42.66	65.83	69.27
9.80	9.33	10.01	9.90	11.76	12.66	15.78	29.99	41.15	47.31	70.71
8.52	9.17	10.47	10.22	11.61	12.36	20.83	31.68	40.15	47.14	71.57
9.36	9.38	9.79	10.15	12.25	14.45	18.07	30.30	36.84	55.30	81.55
8.82	9.16	10.26	10.41	11.88	12.37	16.97	31.48	38.72	55.74	79.04
8.90	9.63	9.56	10.86	12.42	15.79	22.54	33.09	30.39	51.02	74.31
8.87	9.95	9.61	10.48	11.79	12.91	22.40	29.43	41.91	59.39	87.98
9.24	9.31	9.25	10.56	11.15	11.17	16.17	22.45	44.94	66.49	70.17
8.95	9.26	9.69	10.64	10.89	13.44	19.78	36.72	40.54	64.89	85.34
8.63	9.10	9.95	11.74	12.95	13.39	18.84	22.44	39.37	61.87	64.54
9.29	9.67	10.94	11.68	11.15	13.29	15.99	37.38	45.14	68.61	76.18
9.06	10.22	10.31	10.72	12.56	13.98	17.31	30.90	39.09	48.31	70.32
8.78	9.37	9.75	10.62	11.49	14.74	16.98	31.53	47.47	52.20	63.16
8.96	9.04	10.02	9.98	11.46	12.79	25.53	25.40	47.54	51.91	65.76
8.81	9.71	10.56	9.92	11.56	13.10	19.72	20.47	48.55	56.49	61.36
8.45	9.63	9.96	10.30	12.33	16.06	14.00	24.68	41.55	60.94	92.89
9.40	10.09	9.55	10.56	13.08	17.30	16.17	31.47	49.16	48.62	70.63
9.17	9.47	9.97	10.95	11.95	12.57	19.02	22.41	39.82	57.03	68.78
9.69	10.02	9.93	10.35	11.22	14.17	15.44	31.84	46.66	56.77	57.38
9.45	9.33	10.63	10.85	11.44	13.78	23.77	23.93	37.93	54.56	66.00
9.45	9.17	9.48	11.18	11.90	15.37	15.34	25.16	42.61	54.76	74.77
8.76	9.40	10.16	10.36	13.36	13.25	21.35	33.72	39.32	56.45	84.37
8.56	10.04	9.60	11.07	10.80	14.96	15.46	29.19	36.27	57.83	78.77
9.27	9.86	10.54	10.51	11.55	16.80	16.56	27.28	47.12	57.88	65.14
9.26	9.57	10.80	10.04	11.59	15.45	18.85	30.15	33.18	56.13	68.27
8.90	8.89	10.52	10.53	11.67	14.79	17.32	42.66	28.71	64.17	74.35
8.75	9.30	9.90	11.03	11.44	13.45	19.14	28.91	36.40	55.94	62.83
9.04	9.62	10.37	10.79	11.45	12.23	19.74	23.00	44.66	57.62	71.86
8.29	9.93	9.88	10.56	12.42	14.94	17.42	26.82	45.38	57.51	68.95
8.99	9.43	10.46	10.37	13.09	12.01	15.32	24.16	48.92	62.51	69.57
9.31	9.50	10.63	10.50	10.79	14.43	15.33	25.56	40.63	59.75	76.37
9.37	9.69	10.48	10.03	11.27	16.97	17.61	32.00	42.94	60.75	62.90
9.19	9.59	9.39	10.39	11.61	12.73	26.09	30.73	30.48	70.90	75.75
8.58	9.01	11.01	10.76	11.50	12.44	20.62	27.07	31.38	57.03	59.41
8.69	9.69	10.17	11.42	11.24	14.81	15.30	33.41	41.20	49.29	70.01
8.23	9.32	9.85	11.01	11.37	14.25	16.09	41.17	40.15	70.18	71.10
9.00	9.94	9.88	10.41	12.91	13.21	16.90	22.94	41.66	56.68	66.65

9.67	9.94	10.48	10.49	12.37	12.88	16.53	22.40	36.42	59.34	67.63
8.69	9.34	9.76	11.12	13.18	13.57	24.00	19.01	41.52	58.05	60.98
8.91	9.32	10.13	10.40	11.63	13.92	14.99	23.75	35.59	55.15	74.58
9.07	9.67	10.47	10.63	13.04	13.17	14.51	40.64	40.06	57.64	77.22
8.89	9.58	10.38	10.87	11.28	15.11	18.01	31.00	39.45	60.61	85.08
9.46	9.35	9.93	10.50	11.87	14.42	16.60	28.11	41.45	60.23	72.35
9.63	9.64	10.07	10.27	12.03	11.62	20.33	23.23	35.35	54.44	91.39
9.02	10.16	10.28	10.49	12.63	13.60	17.70	36.78	25.70	70.51	68.77
9.38	9.11	11.49	10.23	12.04	13.17	16.66	38.71	33.35	54.20	66.41
8.67	9.28	10.19	11.44	12.31	12.59	16.26	30.91	36.88	49.26	72.08
9.20	9.33	10.65	10.86	11.47	11.95	16.34	32.08	54.15	51.69	83.88
9.50	8.75	9.49	10.76	12.65	13.46	19.90	25.05	40.85	45.32	79.31

Table A-7 ACO waiting queues with LT/T=0.5

1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
1.30	1.48	1.68	2.02	2.21	2.36	2.75	4.51	4.69	7.33	8.59
1.31	1.45	1.75	2.12	2.33	2.45	3.37	3.33	5.21	6.24	8.84
1.22	1.45	1.61	1.76	2.22	2.52	3.35	3.18	6.56	8.32	9.90
1.36	1.52	1.76	1.78	2.24	2.25	2.78	3.44	5.64	6.52	10.45
1.28	1.48	1.54	1.71	2.24	2.39	3.48	2.89	5.66	8.61	8.24
1.22	1.44	1.71	2.12	2.20	2.52	2.81	3.38	4.82	7.22	8.80
1.23	1.43	1.81	1.81	2.37	2.46	2.54	3.58	4.51	6.12	10.23
1.16	1.51	1.71	1.80	2.17	2.53	3.00	4.16	6.30	7.23	8.54
1.32	1.44	1.73	2.07	2.20	2.59	2.82	3.60	7.05	8.24	11.79
1.40	1.42	1.77	2.00	2.18	2.16	3.13	3.72	5.62	6.72	8.59
1.22	1.47	1.86	2.17	2.11	2.49	3.31	3.05	6.56	5.87	8.25
1.36	1.45	1.71	1.89	2.01	2.42	2.76	3.45	4.70	7.91	10.44
1.24	1.49	1.68	1.92	2.11	2.44	2.64	4.69	5.29	8.11	10.54
1.24	1.50	1.77	2.00	2.20	2.61	2.78	3.31	5.02	6.66	7.76
1.21	1.49	1.75	1.97	2.15	2.20	3.05	3.27	5.09	6.08	6.76
1.30	1.36	1.77	1.77	2.21	2.46	3.29	3.73	6.25	6.14	9.54
1.24	1.47	1.56	1.84	2.25	2.21	2.71	3.41	5.99	4.52	9.06
1.35	1.45	1.79	1.82	2.33	2.57	3.35	3.38	5.09	7.69	8.72

1.35	1.40	1.65	1.97	2.19	2.59	3.01	3.94	6.62	6.33	10.47
1.36	1.54	1.75	1.84	2.12	2.48	3.35	2.67	4.54	7.39	9.52
1.26	1.49	1.62	1.95	2.43	2.39	2.45	4.22	5.89	8.31	9.25
1.27	1.52	1.79	2.03	2.20	2.76	2.98	3.48	6.16	7.35	10.91
1.23	1.43	1.70	2.10	2.13	2.78	3.15	3.92	6.18	6.85	9.39
1.24	1.48	1.65	1.91	2.18	2.68	2.83	2.79	5.72	7.39	9.16
1.23	1.39	1.73	1.95	2.29	2.33	2.40	3.86	5.11	6.43	11.00
1.25	1.51	1.55	2.00	2.36	2.29	3.14	3.96	5.66	5.39	9.37
1.33	1.50	1.66	1.99	2.31	2.09	3.15	3.24	4.36	8.21	11.13
1.33	1.44	1.49	1.95	2.19	2.25	3.02	3.51	4.29	6.54	8.23
1.15	1.40	1.78	1.98	2.31	2.40	3.49	3.39	5.21	6.23	8.42
1.34	1.52	1.74	1.86	2.38	2.39	2.33	3.68	4.83	8.34	9.15
1.24	1.53	1.61	1.92	2.11	2.12	2.69	3.45	5.97	6.82	8.51
1.26	1.48	1.62	1.89	2.23	2.45	3.42	4.48	5.71	7.26	7.76
1.37	1.48	1.64	2.03	2.35	2.35	3.16	4.27	7.11	7.31	10.45
1.22	1.53	1.66	1.91	2.25	2.58	3.35	4.11	5.89	8.75	7.64
1.34	1.53	1.67	1.71	2.34	2.56	3.34	4.01	5.27	8.68	8.88
1.17	1.45	1.75	1.97	2.25	2.44	3.01	2.86	6.64	7.50	10.94
1.30	1.50	1.58	2.03	2.27	2.30	2.40	3.13	4.50	7.58	8.38
1.23	1.51	1.64	1.83	2.30	2.45	2.58	3.64	5.48	7.62	8.54
1.28	1.42	1.74	2.19	2.06	2.50	2.73	4.40	5.34	7.00	9.41
1.17	1.49	1.61	1.89	2.12	2.28	2.91	3.36	6.30	7.46	10.11
1.23	1.46	1.79	1.81	2.45	2.38	2.70	3.44	6.45	8.02	8.91
1.21	1.42	1.74	1.77	2.22	2.09	3.21	4.19	5.95	7.16	10.57
1.37	1.51	1.74	1.96	2.13	2.74	2.61	3.54	6.37	6.66	8.75
1.31	1.50	1.63	1.85	2.07	2.53	3.14	4.37	5.54	6.98	8.60
1.32	1.42	1.75	1.90	2.13	2.19	2.85	3.47	4.15	6.87	9.91
1.26	1.47	1.70	1.89	2.29	2.54	2.63	4.03	5.44	6.91	9.52
1.32	1.55	1.58	1.95	2.23	2.30	3.02	3.47	5.42	9.56	9.52
1.24	1.45	1.76	1.99	2.39	2.42	3.38	3.46	5.46	7.67	8.85
1.28	1.42	1.86	1.79	2.31	2.41	2.85	3.58	6.10	5.56	10.70
1.24	1.45	1.80	2.08	2.25	2.65	2.75	2.64	6.04	7.97	8.59

Table A-8 FAC waiting queues with LT/T=0.5

1600	1800	2000	2200	2400	2600	2800	3000	3200	3400	3600
1.41	1.59	1.69	1.91	2.03	2.28	3.73	5.82	6.11	12.16	18.88
1.47	1.55	1.68	1.90	2.38	3.05	3.82	5.53	8.31	12.05	15.28
1.45	1.54	1.73	1.83	2.18	2.35	3.36	5.41	7.24	11.07	15.57
1.38	1.58	1.69	1.95	2.14	2.41	3.34	7.65	7.36	9.54	15.97
1.40	1.51	1.62	1.90	2.24	2.82	3.47	5.99	6.31	11.41	16.68
1.41	1.53	1.67	1.91	2.18	2.39	3.29	5.41	7.28	11.11	17.35
1.33	1.58	1.69	1.91	2.15	2.75	4.01	6.56	5.33	11.46	15.50
1.42	1.53	1.70	1.84	2.23	2.41	4.06	5.28	7.74	12.59	18.22
1.37	1.61	1.60	1.90	2.03	2.18	3.23	5.39	7.44	13.12	14.55
1.34	1.54	1.76	1.89	2.19	2.47	3.55	6.14	7.75	11.91	17.03
1.33	1.42	1.66	1.97	2.20	2.54	3.15	4.99	7.51	12.26	15.39
1.40	1.59	1.85	2.03	2.11	2.53	3.95	5.86	8.21	13.21	17.55
1.42	1.56	1.76	1.99	2.24	2.72	3.03	5.41	7.51	10.57	15.11
1.38	1.53	1.63	1.93	2.16	2.78	3.08	6.22	8.76	10.97	15.22
1.39	1.50	1.70	1.86	2.21	2.44	3.87	4.37	8.98	10.29	16.80
1.34	1.51	1.79	1.83	2.18	2.48	3.67	4.80	9.22	11.80	15.22
1.37	1.55	1.65	1.90	2.33	2.78	3.18	4.64	7.67	12.20	17.91
1.44	1.52	1.64	1.93	2.27	3.01	2.90	5.30	9.59	11.06	15.99
1.33	1.55	1.71	1.89	2.11	2.35	3.40	5.20	7.20	11.22	17.34
1.43	1.55	1.71	1.87	2.19	2.73	3.03	6.02	8.42	11.26	15.41
1.43	1.53	1.72	1.94	2.22	2.64	3.53	5.10	7.31	11.54	16.02
1.43	1.54	1.59	1.91	2.17	2.77	3.36	5.66	7.89	11.82	15.36
1.42	1.55	1.61	1.81	2.34	2.53	3.39	6.41	7.14	11.67	16.58
1.33	1.61	1.60	1.96	2.14	2.94	3.49	6.50	7.13	11.47	16.42
1.43	1.57	1.75	1.84	2.22	3.08	4.06	6.31	8.47	11.27	14.66
1.40	1.59	1.82	1.89	2.17	2.91	3.38	6.16	5.77	11.85	14.72
1.37	1.54	1.80	1.92	2.21	2.77	3.39	8.66	5.20	12.60	18.14
1.38	1.53	1.75	1.91	2.19	2.51	3.48	5.61	6.86	12.22	16.40
1.39	1.49	1.72	1.95	2.12	2.42	3.18	4.89	8.30	11.73	15.60
1.37	1.60	1.71	1.88	2.25	2.88	3.63	6.23	8.11	12.24	16.70
1.38	1.52	1.76	1.92	2.30	2.38	3.12	4.33	9.25	12.62	15.51
1.41	1.53	1.79	1.83	2.10	2.66	3.20	5.44	7.66	11.84	18.00
1.40	1.53	1.83	1.85	2.15	2.72	3.62	5.23	8.18	12.63	13.96
1.33	1.55	1.58	1.86	2.19	2.46	4.22	5.99	5.98	12.77	18.33
1.40	1.51	1.88	1.93	2.11	2.33	3.60	5.12	5.94	11.61	13.89

1.39	1.53	1.72	1.95	2.18	2.86	3.18	5.81	7.48	10.51	15.33
1.34	1.51	1.65	1.93	2.09	2.75	3.26	5.56	6.79	13.70	15.22
1.36	1.55	1.66	1.93	2.28	2.61	3.49	4.71	8.05	12.53	15.44
1.47	1.54	1.84	1.82	2.21	2.44	3.04	4.89	6.43	12.11	16.17
1.37	1.53	1.61	1.97	2.28	2.46	3.52	4.23	8.00	12.26	13.96
1.38	1.52	1.69	1.97	2.21	2.44	3.12	5.28	5.70	11.25	15.02
1.38	1.56	1.78	1.85	2.30	2.49	2.86	6.63	6.52	11.62	15.99
1.37	1.58	1.82	1.95	2.07	2.84	3.29	6.07	7.31	12.05	17.48
1.41	1.57	1.69	1.91	2.16	2.62	3.97	4.85	7.84	12.62	16.28
1.39	1.52	1.73	1.91	2.28	2.25	3.96	5.02	6.83	11.66	18.45
1.43	1.59	1.69	1.91	2.35	2.62	3.58	6.23	5.43	13.87	15.35
1.33	1.49	1.81	1.90	2.23	2.54	3.46	6.33	6.27	11.69	16.29
1.39	1.57	1.73	1.99	2.32	2.53	3.48	7.03	7.24	10.59	15.60
1.41	1.54	1.81	1.87	2.21	2.29	3.18	6.27	10.22	11.81	18.39
1.33	1.52	1.65	2.01	2.20	2.48	3.83	6.01	7.13	10.62	16.67

Appendix B Matlab Code

```
%%start_twostep.m (10/04)
% clear
% clc

global volume_1 volume_2 hw redtime;          %gobal variable
volume_1=283;                                %volume_1 is left turn
volume_2=566;                                %volume_2 is forward lane

l = [volume_1/3600,volume_2/3600,volume_1/3600,volume_2/3600,...
     volume_1/3600,volume_2/3600,volume_1/3600,volume_2/3600]';
                                     %lambda, vehicle arrival rate

hw= 2;                                       %space time between each cars on internal
                                     %movements

redtime= 2;                                %all red time for interections

NumCars_L= fix(volume_1* 1);                %number of cars
generated                                     %watch out the ration
NumCars_F= fix(volume_2* 1);                %number of cars generated

gencar_twostep()

inctime_1018()                              %ACO
Process                                     %Normal Process
% incime_twostep_plot_2()                    %ACO
Process                                     %For PLOT
```

```

%%gencar_twostep.m(10/5)

%arrival time
arrival_time_L = zeros(4,NumCars_L);      %arrival time
arrival_time_F = zeros(4,NumCars_F);      %arrival time
arrival_time = zeros(8,NumCars_F);

%departure time
departure_time = zeros(8,NumCars_F);      %departure time

%delay time
delay_time = zeros(8,NumCars_F);          %delay time

%first, generate spacing between cars
arrival_time_L = max(2,hw-(1/(volume_1/3600)-
hw)*log(rand(4,NumCars_L))); %eq(3.4)
arrival_time_F = max(2,hw-(1/(volume_2/3600)-
hw)*log(rand(4,NumCars_F))); %eq(3.4)

%then, add spacing to get arrival times
for j= 2:NumCars_L
    arrival_time_L(1:4,j)= arrival_time_L(1:4,j-1)+arrival_time_L(1:4,j);
end

for j= 2:NumCars_F
    arrival_time_F(1:4,j)= arrival_time_F(1:4,j-1)+arrival_time_F(1:4,j);
end

% get arrival_time
arrival_time(1,1:NumCars_L)=arrival_time_L(1,:);
arrival_time(2,:)=arrival_time_F(1,:);
arrival_time(3,1:NumCars_L)=arrival_time_L(2,:);
arrival_time(4,:)=arrival_time_F(2,:);
arrival_time(5,1:NumCars_L)=arrival_time_L(3,:);
arrival_time(6,:)=arrival_time_F(3,:);
arrival_time(7,1:NumCars_L)=arrival_time_L(4,:);
arrival_time(8,:)=arrival_time_F(4,:);

arrival_time;
xlswrite('arrivaltime3400',arrival_time');    %excel
file                                           %for excel

```

```

%%inctime_twostep.m(10/30)

light=zeros(8,6); %Traffic signal lights

%the 1st route is 1+5>2+5>2+6>3+7>4+7>4+8
light_1=[1 0 0 0 1 0 0 0]'; %1,5
light_2=[0 1 0 0 1 0 0 0]'; %2,5
light_3=[0 1 0 0 0 1 0 0]'; %2,6
light_4=[0 0 1 0 0 0 1 0]'; %3,7
light_5=[0 0 0 1 0 0 1 0]'; %4,7
light_6=[0 0 0 1 0 0 0 1]'; %4,8
light = {light_1,light_2,light_3,light_4,light_5,light_6};

%ACO parameters
iterations=75; %Number of iterations before choosing a switch time
numants=25; %number of ants used
N=10; %number of ants used in rank-based
evaporation =0.4; %pheromone evaporation rate
local=1; %1 if local search is to be used, 0 otherwise
locstep = 5; %How often local search is performed
localneighbor=4; %Size of local search neighborhood
heur=1; %1 if heuristic information is to be used
elitist=1; %1 if elitist ant is to be used
e=10; % elitist weight
rank=0; %1 if rank-based update is to be used
c=5; %constant c
alpha=1; %constant alpha
beta=1; %constant beta
trials=fix(100/6); %how many trials

totalsig = zeros(1,400); %array containing signal switch times
totalqueue = zeros(8,400); %array containing queue on each movement
%at switch times

await=zeros(8,NumCars_F); %array containing delay of each vehicle

t=0; %start time
initial = zeros(8,1); %number of cars in each direction at
%start of each phase

totalsig(1) = t; %totalsig is vector of switching times
totalqueue(:,1)=initial; %Array of queue length at each change
%in each direction

totalwait=zeros(8,1); %Array of vehicle wait times
next=ones(8,1); %Index of next arrival

NN = zeros(8,1); %how many times signal
switch % add on 4/14
QQ = zeros(8,1); %total queue on each lane

currentwait=zeros(8,1); %The current wait of vehicles waiting
%at the queue

```

```

SDSDT=zeros(8,6);           %Sum Delat Time
SDSCN=zeros(8,6);           %Sum Car Number
Average_Delay=0;             %Average Delay Time

normalized_pher=zeros(iterations,6); %Normalize pheromone on optimal path
average_pher=zeros(trials,iterations); %the average of each trial

figure(3);                   % Rate of convergence with X
ants,evp=X                    %for plot

% green signal for E/W
green_1_5=[1,0,0,0,1,0,0,0]'; %green signal for 1&5
green_1_6=[1,0,0,0,0,1,0,0]'; %green signal for 1&6
green_2_5=[0,1,0,0,1,0,0,0]'; %green signal for 2&5
green_2_6=[0,1,0,0,0,1,0,0]'; %green signal for 2&6
green_E_W=[green_1_5,green_1_6,green_2_5,green_2_6];

% green signal for N/S
green_3_7=[0,0,1,0,0,0,1,0]'; %green signal for 3&7
green_3_8=[0,0,1,0,0,0,0,1]'; %green signal for 3&8
green_4_7=[0,0,0,1,0,0,1,0]'; %green signal for 4&7
green_4_8=[0,0,0,1,0,0,0,1]'; %green signal for 4&8
green_N_S=[green_3_7,green_3_8,green_4_7,green_4_8];

green_T={green_1_5,green_1_6,green_2_5,green_2_6,...
          green_3_7,green_3_8,green_4_7,green_4_8};
greendir_T=[1 1 2 2 3 3 4 4;...
            5 6 5 6 7 8 7 8];
inc=1;                                           %number of signal transitions

step=3;
fgc=1;
kuoco=1;
realmove=[3 2 2 3 3 2 2 3];
dennA=zeros(1,3);

gc=1;                                           %green control(initial=1)
EWNS=0;                                         %E/W=0,N/S=1 control
Roundcontrol=0;
og=[4 3 2 1 8 7 6 5]; %opposite green(next green)
opp=[4 3 2 1 8 7 6 5];
Totalinital=zeros(4,1);

Table{1}=[1 3 4;1 2 4;1 4 -1];
Table{2}=[2 1 3;2 4 3;2 1 3];
Table{3}=[3 4 2;3 1 2;3 4 2];
Table{4}=[4 3 1;4 2 1;4 1 -1];
Table{5}=[5 7 8;5 6 8;5 8 -1];
Table{6}=[6 5 7;6 8 7;6 5 7];
Table{7}=[7 8 6;7 5 6;7 8 6];
Table{8}=[8 7 5;8 6 5;8 5 -1];

```

```

light{1}=[1 0 0 0 1 0 0 0]'; %1,5
ss=1;
% for plot
% s=1;
SDNN=1; % for Sum Delay Time & Sum Car Number

% while t<60*30 %time less than
30mins % lesss than 30 mins
while
t<60*20 %les
s than 20 mins on 4/16

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Chose route

if Roundcontrol == 0 % still in the same round and decide next gc
    stepEWNS=EWNS;
    if step==3
        Roundcontrol=1;

    elseif step==2
        gc=Table{fgc}(1,3); %gc
        step=step+1;
        stephelp=step;

    elseif step==1
        if realmove(fgc)==3
            kuoco=1;
        elseif realmove(fgc)==2
            kuoco=-1;
        end

        if EWNS==0

            dennA(1)=Table{fgc}(1,2); %dennA(1)=3
            dennisa1=greendir_T(1,dennA(1)); %2
            dennisa2=greendir_T(2,dennA(1)); %5

            dennA(2)=Table{fgc}(2,2); %dennA(2)=2
            dennisa3=greendir_T(1,dennA(2)); %1
            dennisa4=greendir_T(2,dennA(2)); %6

            dennA(3)=Table{fgc}(3,2); %dennA(3)=4
            dennisa5=greendir_T(1,dennA(3)); %2
            dennisa6=greendir_T(2,dennA(3)); %6

            [maxgc, argmax]=max([initial(dennisa1)+initial(dennisa2),...
                initial(dennisa3)+initial(dennisa4),kuoco*(initial(dennisa5)...
+initial(dennisa6)),initial(3)+initial(7),initial(3)+initial(8),...
initial(4)+initial(7),initial(4)+initial(8))]);

```

```

if argmax==1 || argmax==2 || argmax==3
    gc=dennA(argmax);
    step=step+1;
    stephelp=step;
    if gc==Table{fgc}(1,3)
        step=step+1;
        stephelp=step;
    end
else %argmax=4|5|6|7
    gc=argmax+1;
    fgc=gc;
    EWNS=2;
    stepEWNS=EWNS;
    step=1;
    stephelp=step;
end

else %EWNS=1
    dennA(1)=Table{fgc}(1,2); %dennA(1)=3
    dennisa1=greendir_T(1,dennA(1)); %2
    dennisa2=greendir_T(2,dennA(1)); %5

    dennA(2)=Table{fgc}(2,2); %dennA(2)=2
    dennisa3=greendir_T(1,dennA(2)); %1
    dennisa4=greendir_T(2,dennA(2)); %6

    dennA(3)=Table{fgc}(3,2); %dennA(3)=4
    dennisa5=greendir_T(1,dennA(3)); %2
    dennisa6=greendir_T(2,dennA(3)); %6

    [maxgc,argmax]=max([initial(dennisa1)+initial(dennisa2),...
    initial(dennisa3)+initial(dennisa4),...
    kuoco*(initial(dennisa5)+initial(dennisa6)),initial(1)+initial(5)...
    ,initial(1)+initial(6),initial(2)+initial(5),initial(2)+initial(6
    )]);

    if argmax==1 || argmax==2 || argmax==3
        gc=dennA(argmax);
        step=step+1;
        stephelp=step;
        if gc==Table{fgc}(1,3)
            step=step+1;
            stephelp=step;
        end
    else %argmax=4|5|6|7
        gc=argmax-3;
        fgc=gc;
        EWNS=0;
        stepEWNS=EWNS;
        step=1;
        stephelp=step;
    end
end

```

```

        if EWNS==2
            EWNS=1;
            stepEWNS=EWNS;
        end

    end
end

%     end
% end
%

if Roundcontrol==1;
    if EWNS==0        %E/W
        [maxgc, argmax]=max([initial(3)+initial(7), initial(3)+initial(8)...
            , initial(4)+initial(7), initial(4)+initial(8)]);
        gc = argmax+4;        %gc
        fgc=gc;
        Roundcontrol=0;
        EWNS=2;
        step=1;
        stephelp=step;

    else                %EWNS=1, N/S
        [maxgc, argmax]=max([initial(1)+initial(5), initial(1)+initial(6)...
            , initial(2)+initial(5), initial(2)+initial(6)]);
        gc = argmax;        %gc
        fgc=gc;
        Roundcontrol=0;
        EWNS=0;
        stepEWNS=EWNS;
        step=1;
        stephelp=step;

    end

    if EWNS==2
        EWNS=1;
        stepEWNS=EWNS;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% stephelp=1
if stephelp==1
    stepgc(1)=gc;
    stepfgc=gc;
    light{1}=green_T{stepgc(1)};    % for ACO light{1}
    if realmove(gc)==2
        kuoco=-1;
    else
        kuoco=1;
    end
end

```



```

if stepEWNS==0

dennA(1)=Table{gc}(1,2);          %dennA(1)=3
dennisa1=greendir_T(1,dennA(1)); %2
dennisa2=greendir_T(2,dennA(1)); %5

dennA(2)=Table{gc}(2,2);          %dennA(2)=2
dennisa3=greendir_T(1,dennA(2)); %1
dennisa4=greendir_T(2,dennA(2)); %6

dennA(3)=Table{gc}(3,2);          %dennA(3)=4
dennisa5=greendir_T(1,dennA(3)); %2
dennisa6=greendir_T(2,dennA(3)); %6

[maxgc, argmax]=max([initial(dennisa1)+initial(dennisa2),...
    initial(dennisa3)+initial(dennisa4),...

kuoco*(initial(dennisa5)+initial(dennisa6)),initial(3)+initial(7),initial(3)+
initial(8),...
initial(4)+initial(7),initial(4)+initial(8)]);

if argmax==1 || argmax==2 || argmax==3
    stepgc(2)=dennA(argmax);
    light{2}=green_T{stepgc(2)}; % for ACO light{2}

        if stepgc(2)~=Table{gc}(1,3)
            stepgc(3)=Table{gc}(1,3);          %step=3
            light{3}=green_T{stepgc(3)}; % for ACO light{3}
            %%%%% step=step+1;

        else % if stepgc(2)==Table{gc}(1,3)

[ maxgc, argmax]=max([initial(3)+initial(7),initial(3)+initial(8)...
    ,initial(4)+initial(7),initial(4)+initial(8)]);
    stepgc(3)= argmax+4;          %step=1'
    light{3}=green_T{stepgc(3)}; % for ACO light{3}
    end

else
    stepgc(2)=argmax+1;
    light{2}=green_T{stepgc(2)}; % for ACO light{2}
    stepgc(3)=opp(stepgc(1));
    light{3}=green_T{stepgc(3)}; % for ACO light{3}
end

else %stepEWNS=1

dennA(1)=Table{stepfgc}(1,2);          %dennA(1)=3
dennisa1=greendir_T(1,dennA(1)); %2
dennisa2=greendir_T(2,dennA(1)); %5

```

```

dennA(2)=Table{stepfgc}(2,2);      %dennA(2)=2
dennisa3=greendir_T(1,dennA(2));  %1
dennisa4=greendir_T(2,dennA(2));  %6

dennA(3)=Table{stepfgc}(3,2);      %dennA(3)=4
dennisa5=greendir_T(1,dennA(3));  %2
dennisa6=greendir_T(2,dennA(3));  %6

[maxgc, argmax]=max([initial(dennisa1)+initial(dennisa2),...
initial(dennisa3)+initial(dennisa4),kuoco*(initial(dennisa5)...
+initial(dennisa6)),initial(1)+initial(5),initial(1)+initial(6),...
initial(2)+initial(5),initial(2)+initial(6)]);

    if argmax==1 || argmax==2 || argmax==3
        stepgc(2)=dennA(argmax);
        light{2}=green_T{stepgc(2)}; % for ACO light{2}

        if stepgc(2)~=Table{stepfgc}(1,3)
            stepgc(3)=Table{stepfgc}(1,3); %step=3
            light{3}=green_T{stepgc(3)}; % for ACO light{3}
        else
            [maxgc, argmax]=max([initial(1)+initial(5),initial(1)+initial(6)...
,initial(2)+initial(5),initial(2)+initial(6)]);
            stepgc(3)= argmax; %step=1'
            light{3}=green_T{stepgc(3)}; % for ACO light{3}
        end

    else %argmax=4|5|6|7
        stepgc(2)=argmax-3;
        light{2}=green_T{stepgc(2)}; % for ACO light{2}
        stepgc(3)=opp(stepgc(1));
        light{3}=green_T{stepgc(3)}; % for ACO light{3}
    end
end
end

if stephelp==2
    stepgc(1)=gc;
    light{1}=green_T{stepgc(1)}; % for ACO light{1}
    if Table{fgc}(1,3)~=gc
        stepgc(2)=Table{fgc}(1,3);
        light{2}=green_T{stepgc(2)}; % for ACO light{2}
        steprime=1;
    else %if Table{fgc}(1,3)=gc
        if stepEWNS==0

            [maxgc, argmax]=max([initial(3)+initial(7),initial(3)+initial(8)...
,initial(4)+initial(7),initial(4)+initial(8)]);
            stepgc(2) = argmax+4;
            light{2}=green_T{stepgc(2)}; % for ACO light{2}

```

```

        steprime= 2;
%
step=step+1;                                %%
step=step+1;
        else %stepEWNS==1

[ maxgc, argmax]=max([ initial(1)+initial(5), initial(1)+initial(6)...
                    , initial(2)+initial(5), initial(2)+initial(6)]);
        stepgc(2) = argmax;
        light{2}=green_T{stepgc(2)}; % for ACO light{2}
        steprime= 2;
%
step=step+1;                                %%
step=step+1;
        end
    end

    if steprime==1
        if stepEWNS==0

[ maxgc, argmax]=max([ initial(3)+initial(7), initial(3)+initial(8)...
                    , initial(4)+initial(7), initial(4)+initial(8)]);
        stepgc(3) = argmax+4;
        light{3}=green_T{stepgc(3)}; % for ACO light{3}
        else
            %stepEWNS==1

[ maxgc, argmax]=max([ initial(1)+initial(5), initial(1)+initial(6)...
                    , initial(2)+initial(5), initial(2)+initial(6)]);
        stepgc(3) = argmax;
        light{3}=green_T{stepgc(3)}; % for ACO light{3}
        end
    else
        %steprime==2
%
step=step+1;                                %%
step=step+1;
        stepgc(3)=opp(stepgc(1));
        light{3}=green_T{stepgc(3)}; % for ACO light{3}

    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% step=3
if stephelp==3
    stepgc(1)=gc;
    light{1}=green_T{stepgc(1)}; % for ACO light{1}
    if stepEWNS==0
        [ maxgc, argmax]=max([ initial(3)+initial(7), initial(3)+initial(8)...
                            , initial(4)+initial(7), initial(4)+initial(8)]);
        stepgc(2)= argmax+4;

```

```

        light{2}=green_T{stepgc(2)}; % for ACO light{2}
    else %stepEWNS==1
        [maxgc, argmax]=max([initial(1)+initial(5),initial(1)+initial(6)...
            ,initial(2)+initial(5),initial(2)+initial(6)]);
        stepgc(2) = argmax;
        light{2}=green_T{stepgc(2)}; % for ACO light{2}
    end

    stepgc(3)=opp(stepgc(1));
    light{3}=green_T{stepgc(3)}; % for ACO light{3}

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ACO_twostep() %run ACO to find optimal signal
T;
sigset(1) = t;
sigset(2) = sigset(1)+gsetting(1);
sigset(3) = sigset(2)+gsetting(2);

final_sigset(1)=sigset(1); %final answer
final_setting(1)=gsetting(1); %final answer

% %prob{1}
% plot(1, prob{1}(final_setting(1)-4-redtime)) %plot
% hold on

t = sigset(2); %Advance time to next signal change

movecars_departuretime() %The chosen path is evaluated and vehicle's
wait %time is computed

% totalsig(inc)=final_sigset(s); %keep track of signal
% totalqueue(:,inc)=initial; %keep track of queue length

inc=inc+1;
1; %ACO_1 is done
end

end

xlswrite('departuretime3400',departure_time') %excel
file %for excel
arrival_time;
departure_time;

```

```

numbercars=0;

for i=1:8
    for j=1:NumCars_F
        if departure_time(i,j) > 0 & departure_time(i,j) <
1800                %(<30 mins)
            delay_time(i,j)=departure_time(i,j)-arrival_time(i,j);
            numbercars=numbercars+1;
        end
    end
end

xlswrite('delaytime3400',delay_time') %excel
file %for excel

Average_Delay= sum(sum(delay_time))/numbercars %Average Delay

Average_Queue=sum(QQ./NN)/8
%add on 4/13

```

```

%%ACO_twostep.m(10/05)

%Pheromone levels are initialized, then ants construct new solutions on
%each iteration

prob = {ones(26,1),ones(26,26)};
prob2=prob; %prob2 used with heuristic

%prob is a array consisting of
%[26x1 double] [26x26 double]

%The matrix in the first slot is for the next time change and
%the matrix in the second slot is for the time change after that. The
%ij-th slot of the 2nd matrix gives the probability of an ant moving
%from time t+i to time t+i+j. It is updated after each iteration

weight = [5:30]';
weight = (exp(-abs(((max(light{1}.*initial)-1)*hw)-...
    (weight+redtime))/c)).^(beta);
%Weight is used for heuristic information

gminwait=10000;

for trial = 1:trials %how many trials for the
plot %for plot
    for iteration = 1:iterations %based on ants
        updatepher_twostep() %Ants create solution and find optimal path
    end

x=1:iterations; %for
plot
    x;
    normalized_pher;
    plot(x,normalized_pher(:,ss));
    title('Rate of convergence with 10ants, evaporation=0.2');
    xlabel('Number of iterations');
    ylabel('Normalized pheromone on optimal path');
    hold on;
end %
for plot

%%updatepher_twostep.m (10/05)

```

```

% At each iteration ants create and evaluate candidate solutions, here
% they deposit pheromone, based on ACO parameters, and pheromone is
% evaporated.

minwait=10000;
if local == 1 && mod(iteration,locstep )==0%Do local search on 5t iteration
    %P.38,39
    locmin(1)=max(1,(gsetting(1)-redtime-4-localneighbor));
    locmax(1)=min(26,(gsetting(1)-redtime-4+localneighbor));
    locmin(2)=max(1,(gsetting(2)-redtime-4-localneighbor));
    locmax(2)=min(26,(gsetting(2)-redtime-4+localneighbor));
    locprob = {ones(1,locmax(1)-locmin(1)+1),ones(locmax(1)-...
        locmin(1)+1,locmax(2)-locmin(2)+1)};

    optcen_twostep()

    antwait =waittime;           %waittime of each ant
    aset = setting;             %signal setting of each ant
    for a = 2:numants           %Number of ants
        optcen_twostep()        %calc waittime for each ant
        antwait = [antwait ;waittime];
        aset = [aset; setting];
    end

else

    optcen2_twostep()

    antwait =waittime;          %wait time of each ant
    aset = setting;             %signal setting of each ant
    for a = 2:numants           %Number of ants
        optcen2_twostep()       %calc wait time for each ant
        antwait = [antwait ;waittime];
        aset = [aset; setting];
    end

end

%Ranking

sorted = sortrows([antwait,aset]);
aset = sorted(:,2:3);

if minwait < gminwait           %Pick best solution so far
    gminwait = minwait;
    goptsig = optsig;
    gsetting = [goptsig(2)-goptsig(1),goptsig(3)-goptsig(2)];
end

if ~(mod(iteration, localneighbor )==0 && local ==1)

```

```

    %If "not" using local search or "not" local search step
    prob = {(1-evaporation)*prob{1},(1-evaporation)*prob{2}};    %Evaporation
of pheromone

    if rank == 1    %Rank-Based Ant System (Rank included Elitist)
        for r = 1:size(aset,1)    %1:number of ants
            for s = 1:size(aset,2)    %1:number of settings
                %Add weight to best solution of last group
                if s ==1
                    xx=aset(r,s)-4-redtime;
                    prob{1}(xx)=prob{1}(xx)+(max(0,(N-r))/(antwait(r)))...
                        ^alpha);
                else %s==2
                    xx=aset(r,s-1)-4-redtime;
                    yy=aset(r,s)-4-redtime;
                    prob{2}(xx,yy)=prob{2}(xx,yy)+(max(0,(N-r))/...
                        (antwait(r)))^alpha);
                end
            end
        end
    end

    else    %Ant System
        for r = 1:size(aset,1)    %1:number of ants
            for s=1:size(aset,2)    %1:number of settings
                %Deposit pheromone on paths
                if s ==1
                    xx=aset(r,s)-4-
redtime;    %
                    prob{1}(xx) = prob{1}(xx)+(1/antwait(r))^alpha);
                else %s=2
                    xx=aset(r,s-1)-redtime-4;
                    yy=aset(r,s)-redtime-4;
                    prob{2}(xx,yy)=prob{2}(xx,yy)+(1/antwait(r))^alpha);
                end
            end
        end
    end

    if elitist ==1    %Elitist Ant System(Rank included Elitist)
        hh=gsetting(1)-redtime-4;
        prob{1}(hh) = prob{1}(hh) + ((e)/ gminwait)^alpha) ;

        gg=gsetting(2)-redtime-4;
        prob{2}(hh,gg) = prob{2}(hh,gg) + ((e)/ gminwait)^alpha) ;
    end

    if heur==1    %Heuristics
        prob2={(prob{1}.*weight),prob{2}};% now only works on 1st layer
        % cuz 2nd layer is no use in
        % result
    else
        prob2=prob;
    end

```



```

        end

end

% % for NO heur value
% normalized_pher(iteration,ss)= max(prob{1})/sum(prob{1}); % normalize
pheromone on optimal
%                                % path

% for heur value
normalized_pher(iteration,ss)= max(prob2{1})/sum(prob2{1}); % normalize
pheromone on optimal
                                % path

```

```

%Each ant creates a candidate solution randomly based on pheromone
%deposits and then the expected delay is computed to evaluate the solution
%This is used on "local search" step

%Randomly pick next solution based on pheromone
sumprob = {sum(locprob{1},1),sum(locprob{2},2)};

setting=zeros(1,2);           %the length of next signal phase
signal=zeros(2,1);

signal(1)=rand*(sumprob{1}(1));
aaa = 1;
while signal(s)>sum(locprob{1}(1:aaa))
    aaa=aaa+1;
end
setting(1)=locmin(1)+ 4 + (aaa-1) + redtime;    %setting(1)

signal(2)=rand*sumprob{2}(aaa);    %s=2
bbb=1;
while signal(2)>sum(locprob{2}(aaa,1:bbb))
    bbb=bbb+1;
end
setting(2)=locmin(2)+ 4 + (bbb-1) + redtime;    %setting(2)

sigset = t;           %sigset is next switch time

% compute sigset
for i=1:size(setting,2)
    sigset=[sigset,sigset(i)+setting(i)];
end

T{1} = initial;
%T{1} is vehicles in queue initially
%T{2} is expected number of vehicles in queue at next signal change
%T{3} is expected number of vehicles in queue at second signal change

%On green movements
T{2} = light{1}.*max(0 , T{1}- floor((setting(1)-redtime)...
    /hw +1)+ 1.* min(hw*(T{1}-1)./(1-1.*hw),setting(1)));
%On red movements
T{2} = T{2} + ~light{1}.*(T{1}+ 1.*setting(1));

%On green movements
T{3} = light{2}.*max(0 , T{2} - floor((setting(2)-redtime)...
    /hw +1)+ 1.* min(hw*(T{2}-1)./(1-1.*hw),setting(2)));
%On red movements
T{3} = T{3} + ~light{2}.*(T{2}+ 1.*setting(2));

compwait_twostep()           %compute expected wait time of signal
if waittime < minwait

```

```

        minwait = waittime;
        optsig = sigset;
        optsetting = setting;
end

%%optcen2_twostep.m (10/02)

%Each ant creates a candidate solution randomly based on pheromone deposits
%and then the expected delay is computed to evaluate the solution
%This is used when "not" local search

sumprob = {sum(prob2{1},1), sum(prob2{2},2)};
%Randomly pick next solution based on pheromone

setting=zeros(1,2);           %the length of next signal phase
signal=zeros(2,1);

signal(1)=rand*(sumprob{1}(1)); %Choose signal switch times based on
                                %pheromone
aa = 1;
while signal(1)>sum(prob2{1}(1:aa))
    aa=aa+1;
end
setting(1) = aa+4+redtime;      %setting(1)

signal(2)=rand*(sumprob{2}(aa));
bb=1;
while signal(2)>sum(prob2{2}(aa,1:bb))
    bb=bb+1;
end
setting(2) = bb+4+redtime;      %setting(2)

sigset=t;                      %sigset is next switch time

% compute sigset
for i=1:size(setting,2)
    sigset=[sigset,sigset(i)+setting(i)];
end

T{1} = initial;
%T{1} is vehicles in queue initially
%T{2} is expected number of vehicles in queue at next signal change
%T{3} is expected number of vehicles in queue at second signal change

%On green movements
T{2} = light{1}.*max(0 , T{1} - floor((setting(1)-redtime)...
    /hw +1)+ 1.* min(hw*(T{1}-1)./(1-l.*hw),setting(1)));
%On red movements
T{2} = T{2} + ~light{1}.*(T{1}+ 1.*setting(1));

```

```

%On green movements
T{3} = light{2}.*max(0 , T{2} - floor((setting(2)-redtime)...
    /hw +1)+ 1.* min(hw*(T{2}-1)./(1-1.*hw),setting(2)));
%On red movements
T{3} = T{3} + ~light{2}.*(T{2}+ 1.*setting(2));

compwait_twostep()                %compute expected wait time of signal
if waittime < minwait
    minwait = waittime;
    optsig = sigset;
    optsetting = setting;
end

```

```

%%compwait_twostep.m(10/05)

```

```

%computes the expected delay time of signal

%computes lower bound on extra wait time
cycle = zeros(8,1);
M = 30/hw+1 - 30.*1; %how many cars can still leave during 30 secs.
q= T{3}+(~light{3})*(5+redtime).*1;
for k =1:8
    while T{3}(k) > M(k) - (~light{3}(k))*(5+redtime)*1(k) + cycle(k)*...
(M(k) - (5+redtime)*1(k)) && cycle(k) <5
        cycle(k) = cycle(k) +1;
        q(k) = T{3}(k) -(M(k) - (~light{3}(k))*(5+redtime)*1(k) + ...
(cycle(k)-1)*(M(k) - (5+redtime)*1(k)));
        if cycle(k) == 5 && iteration ==70
            t;
        end
    end
end
end

%%compute number of "initial vehicles released", "new arrivals released",
%%and "not released"
initrel=zeros(8,2); %initial vehicles released
newrel=zeros(8,2); %new arrivals released
notrel=zeros(8,2); %not released

for i=1:2
    initrel(:,i) = light{i}.* min(1 + fix((setting(i)-redtime)/...
hw),T{i}); %initial vehicles released %eq(3.1)
end

for i=1:2
    newrel(:,i) = light{i}.*max(0,min(1+fix((setting(i)-redtime)...
/hw)-T{i},1.*(T{1}-1)*hw./(1- 1.* hw)));
    %new arrivals released eq(3.11)
end

for i=1:2
    notrel(:,i) = light{i}.*(T{i} - initrel(:,i)); %not released
end

% compute delay time of each group of vehicles
inwait=zeros(8,3);
newwait=zeros(8,3);
notrwait=zeros(8,3);
nnrwait=zeros(8,3);
redwait=zeros(8,3);

for i=1:2
    inwait(:,i) = initrel(:,i).*(initrel(:,i)-1)*hw/2;
end % end%eq(3.6)
inwait(:,3)=sum(inwait(:,1:2),2); %total delay

```

```

for i=1:2
    newwait(:,i) = light{i}.*newrel(:,i)*hw.*(T{i}+1)/2; %eq(3.8)
end
newwait(:,3)=sum(newwait(:,1:2),2); %total delay

for i=1:2
    notrwait(:,i) = light{i}.*notrel(:,i)*setting(i); %eq(q x (t2-t1))
end
notrwait(:,3)=sum(notrwait(:,1:2),2); %total delay

for i=1:2
    nnrwait(:,i) = light{i}.*max(0,1.*(setting(i)-redtime-...
        newrel(:,i)*hw).^2/2.*sign(((T{i}-1)*hw)./(1-1.*hw)-setting(i)+...
        redtime)); %eq(3.12)
end
nnrwait(:,3)=sum(nnrwait(:,1:2),2); %total delay

new = light{3}.*max(0,1.*(q-1)*hw./(1- 1*hw)); %q-1

wait3 = ~light{3}.*T{3}*(5+redtime)+ cycle*(30+5+redtime)^2.*1./2+M.*...
    (M-1)/2.*cycle + max(0,q.*(q-1)*hw/2) + new*hw.*(q+1)/2;

for i=1:2
    redwait(:,i) = ~light{i}.*(T{i}*setting(i) + 1.*(setting(i)...
        .^2)/2); %eq(3.14)
end
redwait(:,3)=sum(redwait(:,1:2),2); %total delay

% Total expect delay time
wait = inwait(:,3) + newwait(:,3) + notrwait(:,3) + nnrwait(:,3) + ...
    redwait(:,3) + currentwait + wait3; %eq(3.15)

% Average expect delay time

waittime =(sum(wait))/(4*(1(1)+1(2))*(sigset(3)-sigset(1))+sum(initial));
%eq(p.35)

```

%%Movecars_departuretime.m(8/26)

```

%Move cars according to the optimal control

setting=sigset(2)-sigset(1);
queue =initial;

for i=1:8
    if light{1}(i)==1
        k=i;                                % k is which line to move car
        NN(k)=NN(k)+1;                      % N is how many times signal
switch      %add 4/13
    QQ(k)=QQ(k)+initial(k);                % total queue on each lane

        movecars_on_green_departuretime() %clear green cars on green
movements
    end
end

for i=1:8
    if light{1}(i)==0
        k=i;                                % k is which line to be red
        movecars_on_red_departuretime()    %clear green cars on red movements
    end
end

%%Movecars_on_green_departuretime.m(8/26)

m=next(k);
n=0;
    for j=1:initial(k)                      %loop through vehicles in queue to
                                            %determine if relased or not
        if hw*(j-1) <= setting(1)-redtime %Vehicle relased
            await(k,m+(j-1))=await(k,m+(j-1))+hw*(j-1);
            departure_time(k,m+(j-1))=sigset(1)+hw*(j-1);
            n=n+1;                          %how many cars have left
        end
    end

end

m=m+n;

next(k)=m;                                %next is now the index of the next vehicle
                                            %to be released

initial(k)=0;

while arrival_time(k,m) <sigset(2)-redtime %new arrivals while
                                            %signal is still green

    if arrival_time(k,m)<sigset(1)+(queue(k)-1)*hw
        %if arrives before traffic has been released

```

```

    if queue(k)*hw <= setting(1)-redtime
        %if initial queue will be released on current signal

        await(k,m)=sigset(1)+hw*queue(k)-arrival_time(k,m);
        departure_time(k,m)=sigset(1)+ hw*queue(k);
        next(k)=next(k)+1;
        queue(k)=queue(k)+1;

    else
        %If queue is too long then calculation begins on next (red)
        %phase

        initial(k)=initial(k)+1;
        queue(k)=queue(k)+1;
    end

    else
        %if vehicle arrives when queue is empty then
        %no wait
        await(k,m)=0;
        departure_time(k,m)=arrival_time(k,m);
        next(k)=m+1;
        end

    m=m+1;
end

if arrival_time(k,next(k)+initial(k))<sigset(2) &&...
    arrival_time(k,next(k)+initial(k))>sigset(2)-redtime
    %cars arrival during redtime window
    initial(k) = initial(k)+1;
end

departure_time;

currentwait(k) = sum(max(0,(-arrival_time(k,next(k):next(k))+...
initial(k))+sigset(2))));

%%Movecars_on_red_departuretime.m(8/26)

m=next(k);
initial(k)=0; %reset initial

while arrival_time(k,m) <sigset(2)
    initial(k)=initial(k)+1; %Count vehicles in %queue
    m=m+1;
end
currentwait(k) = sum(await(k,next(k):next(k)+max(0,(initial(k)-1))));

```



```

%%choose_route_1.m(10/05)

% pick up next E/W traffic route by comparing current wait time
% compare 1+5,2+6,2+5,1+6
[val,ind]=max([currentwait(1)+currentwait(5),currentwait(1)+...
    currentwait(6),currentwait(2)+currentwait(5),currentwait(2)+...
    currentwait(6)]); % maximum currentwait and then release

if ind==1 %1+5
    if (currentwait(2)+currentwait(5))> (currentwait(1)+currentwait(6))
        %if 2+5>1+6
        light{2}=green_2_5;
    else
        light{2}=green_1_6;
    end
    light{1}=green_1_5;
    light{3}=green_2_6;
end

if ind==2 %2+6
    if (currentwait(2)+currentwait(5))> (currentwait(1)+currentwait(6))
        %if 2+5>1+6
        light{2}=green_2_5;
    else
        light{2}=green_1_6;
    end
    light{1}=green_2_6;
    light{3}=green_1_5;
end

if ind==3 %2+5
    if (currentwait(2)+currentwait(6))> (currentwait(1)+currentwait(5))
        %if 2+6>1+5
        light{2}=green_2_6;
    else
        light{2}=green_1_5;
    end
    light{1}=green_2_5;
    light{3}=green_1_6;
end

if ind==4 %1+6
    if (currentwait(2)+currentwait(6))> (currentwait(1)+currentwait(5))
        %if 2+6>1+5
        light{2}=green_2_6;
    else
        light{2}=green_1_5;
    end
    light{1}=green_1_6;
    light{3}=green_2_5;
end

%%Fullact.m(1018)

```

```

%Fully actuated control

t=0; %start time
totalsig = t; %totalsig is vector of switching times
redtime=2;

totalwait=zeros(8,1); %Array of vehicle wait times

gc=1; %green control(initial=1)
EWNS=0; %E/W=0,N/S=1 control
Roundcontrol=0;
og=[4 3 2 1 8 7 6 5]; %opposite green(next green)
Totalinital=zeros(4,1);

green_T = zeros(8,8); %Traffic signal lights

green_1=[1 0 0 0 1 0 0 0]'; %1,5
green_2=[1 0 0 0 0 1 0 0]'; %1,6
green_3=[0 1 0 0 1 0 0 0]'; %2,5
green_4=[0 1 0 0 0 1 0 0]'; %2,6
green_5=[0 0 1 0 0 0 1 0]'; %3,7
green_6=[0 0 1 0 0 0 0 1]'; %3,8
green_7=[0 0 0 1 0 0 1 0]'; %4,7
green_8=[0 0 0 1 0 0 0 1]'; %4,8
green_T = [green_1,green_2,green_3,green_4,green_5,green_6,green_7,green_8];

green=zeros(8,1);

greendir_T=[1 1 2 2 3 3 4 4;5 6 5 6 7 8 7 8]; %Traffic signal light

greendir=zeros(2,1);

nongreendir_T=zeros(6,8);
nongreendir_T=[2 2 1 1 1 1 1 1;3 3 3 3 2 2 2 2;4 4 4 4 4 4 3 3;...
    6 5 6 5 5 5 5 5;7 7 7 7 6 6 6 6;8 8 8 8 8 7 8 7];

reddir_T=[2 2 1 1 4 4 3 3;6 5 6 5 8 7 8 7];

reddir=zeros(2,1);

green=green_T(:,gc);
greendir=greendir_T(:,gc);
reddir=reddir_T(:,gc);

next=ones(8,1); %Index of next arrival

initial = zeros(8,1); %Number of cars in each direction at
%start of optimization
totalqueue=initial; %Array of queue length at each change

```

```

                                %in each direction
inc =1;
chg=zeros(8,1);
currentwait=zeros(8,1);      %The current wait of vehicles waiting
                                %at the queue
nextarrival = ones(8,1);
changegreen=0;
first=1;
fulltime=0;

NN = zeros(8,1);             %how many times signal
switch                       % add on 4/14
QQ = zeros(8,1);             %total queue on each lane

Average_Delay=0;              %Average Delay Time

step=1;
fgc=1;
kuoco=1;
realmove=[3 2 2 3 3 2 2 3];
dennA=zeros(1,3);

% while t<min(arrival_time(:,end))-100
while
t<60*20                      %add
on 4/25
    sigset = [t];
    nextchangefullact_universal()

t=t + redtime;

    if endflag ==1
        for j=1:4
            k=nongreendir_T([j],gc);
            m=nextarrival(k);
            while arrival_time(k,m) <t
                m=m+1;
            end
        end
    end

end

goptsig
=[sigset,t,t+7];              %paramete
r change
sigset = [sigset, goptsig(2),goptsig(3)];

movecars_departuretime()

Table{1}=[1 3 4;1 2 4;1 4 -1];

```

```

Table{2}=[2 1 3;2 4 3;2 1 3];
Table{3}=[3 4 2;3 1 2;3 4 2];
Table{4}=[4 3 1;4 2 1;4 1 -1];
Table{5}=[5 7 8;5 6 8;5 8 -1];
Table{6}=[6 5 7;6 8 7;6 5 7];
Table{7}=[7 8 6;7 5 6;7 8 6];
Table{8}=[8 7 5;8 6 5;8 5 -1];

```

```

if Roundcontrol == 0 % still in the same round and decide next gc
    if step==3
        Roundcontrol=1;

    elseif step==2
        gc=Table{fgc}(1,3);
        step=step+1;

    elseif step==1
        if realmove(fgc)==3
            kuoco=1;
        elseif realmove(fgc)==2
            kuoco=-1;
        end

        dennA(1)=Table{fgc}(1,2); %dennA(1)=3
        dennisa1=greendir_T(1,dennA(1)); %2
        dennisa2=greendir_T(2,dennA(1)); %5

        dennA(2)=Table{fgc}(2,2); %dennA(2)=2
        dennisa3=greendir_T(1,dennA(2)); %1
        dennisa4=greendir_T(2,dennA(2)); %6

        dennA(3)=Table{fgc}(3,2); %dennA(3)=4
        dennisa5=greendir_T(1,dennA(3)); %2
        dennisa6=greendir_T(2,dennA(3)); %6

        [maxgc,argmax]=max([initial(dennisa1)+initial(dennisa2),...
            initial(dennisa3)+initial(dennisa4),...
            kuoco*(initial(dennisa5)+initial(dennisa6))]));

        gc=dennA(argmax);
        step=step+1;

    end
end

```

```

if Roundcontrol==1;
    if EWNS==0 %E/W
        [maxgc,argmax]=max([initial(3)+initial(7),initial(3)+initial(8)...
            ,initial(4)+initial(7),initial(4)+initial(8)]);
        gc = argmax+4;
    end
end

```

```

        fgc=gc;
        Roundcontrol=0;
        EWNS=1;
%        step=1;

    else
        %EWNS=1, N/S
        [maxgc, argmax]=max([initial(1)+initial(5),initial(1)+initial(6)...
            ,initial(2)+initial(5),initial(2)+initial(6)]);
        gc = argmax;
        fgc=gc;
        Roundcontrol=0;
        EWNS=0;
%        step=1;
    end
end

green=green_T(:,gc);
greendir=greendir_T(:,gc);
reddir=reddir_T(:,gc);

    totalsig = [totalsig, t];
    totalqueue=[totalqueue,initial];
    first =0;
    [totalsig;totalqueue];
end

xlswrite('departuretime3200',departure_time');    %excel
file                                             %for excel

arrival_time;
departure_time;
numbercars=0;

for i=1:8
    for j=1:NumCars_F
        if departure_time(i,j) > 0 & departure_time(i,j) <
1800
            %(<30 mins)
            delay_time(i,j)=departure_time(i,j)- arrival_time(i,j);
            numbercars=numbercars+1;
        end
    end
end

xlswrite('delaytime3200',delay_time');           %excel
file                                             %for excel

numbercars;
Average_Delay=sum(sum(delay_time))/numbercars

Average_Queue=sum(QQ./NN)/8
%add on 4/13
%%nextchangefullact_universal.m(10/18)

```

```

%Inputs are current quene
%arrival times
%green
%Determines next time fully actuated controller changes the signal

changegreen=0; %Is 1 if conditions for changing signal are met
endflag =0; %Is 1 if signal should be changed
time =zeros(1,8); %Length of signal
timecontrol=0;

for k =1:2 %clear green cars
    time(greendir(k)) = max(0,(initial(greendir(k))-1))*hw;
end

if max(time) >30 %if time to release cars
in %parameter change %queue is longer than max green time

    mintime = [time(greendir(1)),time(greendir(2))];
    [maxtime,argmax]=max(mintime);
    [mintime,argmin]=min(mintime);

nextarrival(greendir([argmax]))=nextarrival(greendir([argmax]))+16; %parameter change
    while arrival_time(greendir([argmin]),nextarrival(greendir([argmin])))
< t+30 %parameter change
        nextarrival(greendir([argmin]))=nextarrival(greendir([argmin]))+1;
    end

%
time=30; %parameter change %parameter change
timecontrol=30;
endflag=1;

t=t+30; %parameter change %parameter change
ameter change

else
    nextarrival = nextarrival+initial.*green; %All cars in queue are
                                                %released
end

while endflag ~= 1
    endclear = [0,0]; %Is 1 if there are no more cars
                        %arriving

```

```

if max(time) ~=0                %If there are cars in queue
    while and(endclear(1),endclear(2)) ~=1

        if arrival_time(greendir(1),nextarrival(greendir(1)))...
            <t+time(greendir(1))+2 %If car arrives before queue
                                    %is released
            time(greendir(1)) = time(greendir(1))+2;
            nextarrival(greendir(1))=nextarrival(greendir(1))+1;
        else
            endclear(1)=1;
        end

        if arrival_time(greendir(2), nextarrival(greendir(2)))< ...
t+time(greendir(2))+2 %If car arrives before queue is released
            time(greendir(2)) = time(greendir(2))+2;
            nextarrival(greendir(2))=nextarrival(greendir(2))+1;

        else
            endclear(2)=1;
        end

        if max(time)>29                %If cars keep arriving
until    %parameter change            %the max green time

            endflag=1;
            endclear(1)=1;
            endclear(2)=1;

t=t+30;                                %parameter
change

        end

    end

    mintime = [time(greendir(1)),time(greendir(2))]; %Which green
                                                %signal requires less green time
    [mintime,argmin]=min(mintime);
    %Let cars pass on shorter green signal if other direction is
    %still green

    while arrival_time(greendir(argmin),nextarrival(greendir(argmin)))...
        < t+max(time)*(1-endflag)
        nextarrival(greendir(argmin))=nextarrival(greendir(argmin))+1;
    end

else    %If queue is empty
    time = max(time);                %Make time a scalar

    if arrival_time(greendir(1),nextarrival(greendir(1)))< t+2
                                                %If car arrives
                                                %before minimum green if up

```

```

%         time = time+2;
        timecontrol=timecontrol+2;
        nextarrival(greendir(1))=nextarrival(greendir(1))+1;
        if arrival_time(greendir(2),nextarrival(greendir(2)))< t+2
            nextarrival(greendir(2))=nextarrival(greendir(2))+1;

        end
    elseif arrival_time(greendir(2),nextarrival(greendir(2)))< t+2
%         time = time+2;
        timecontrol=timecontrol+2;
        nextarrival(greendir(2))=nextarrival(greendir(2))+1;

    else
        endflag=1;
    end

%         if fulltime+
time >28                                     %parameter change
        if fulltime+
timecontrol >28                             %parameter
change
            endflag =1;
%         t = t +time;
            t = t +timecontrol;
        end
        %end
    end
    fulltime=max(max(time),1)+fulltime; %Check if reached minimum
green    %parameter change
                                                %time
    if fulltime <=
5                                             %parameter change
        endflag=0;
    end

    if endflag ~=1

t=t+max(max(time),1)    ;                    %par
ameter change

    time =0;
    end

    if endflag ==1
        time =max(time);

        if EWNS==0        %E/W

```