Algorithms for the Traffic Light Setting Problem on the Graph Model*

Shiuan-Wen Chen, Chang-Biau Yang† and Yung-Hsing Peng
Department of Computer Science and Engineering
National Sun Yat-sen University, Kaohsiung, Taiwan
†cbyang@cse.nsysu.edu.tw

Abstract

As the number of vehicles increases rapidly, traffic congestion has become a serious problem in a city. The *traffic light setting* problem is to investigate how to set the given traffic lights such that the total waiting time of vehicles on the roads is minimized. In this paper, we use a graph model to represent the traffic network. On this model, some characteristics of the setting problem are presented and analyzed. We first devise a branch and bound algorithm for obtaining the optimal solution of the traffic light setting problem. In addition, the *genetic algorithm* (GA), the *particle swarm optimization* (PSO) and the *ant colony optimization* (ACO) algorithm are also adopted to get the near optimal solution. Then, to extend this model, we add the assumption that each vehicle can change its direction. In our experiments, we also transform the map of Kaohsiung city into our graph model and test each algorithm on this graph. Our results show that GA seems to be a good strategy for setting traffic lights.
Key words: graph model, traffic light, algorithm

1 Introduction

In statistical physics, physicists have attempted to understand the fundamental principles of traffic flow for almost half a century. Recently, they developed some *cellular automata models* (CA) [1, 2, 5, 9, 14, 3] which provides a theoretical framework for modeling the traffic flow. The first cellular automata model is the *Biham, Middleton, and Levine model* (BML model) [1, 5], which is composed of simple two-dimensional lattices. Each cell of the lattice represents an eastbound street or a northbound street. Each cell may be either empty, or occupied by an eastbound vehicle or a northbound vehicle. There is only one traffic light in each cell, and the cycle time of the traffic light is equal to one unit of time step. In addition to the original CA model, another probabilistic CA model, the *Nagel-Schreckenberg* (NaSch) [14] model, was also proposed for simulating the one-dimensional highway traffic. It is thought to be the simplest CA model where the movement of all vehicles is implemented by a set of simple rules, including acceleration, slowing down, randomization and car motion [14]. To make it more accurate, Chowdhury and Schadschneider [3] proposed the *ChSch* model, which combines basic ideas from the BML model of the city traffic and the NaSch model of the highway traffic. Besides, Brockfeld *et al.* [2] also studied the influence of various kinds of signal control strategies on the ChSch model. In the past few years, various modifications and extensions for these models [2, 9] have also been proposed to improve their reality.

Some researchers [8, 7, 9] tried to focus on an isolated urban intersection. In general, an isolated intersection is a basic unit of the traffic network, which is believed to be contributive to the understanding of the global optimization in the traffic network. Many strategies for traffic light setting [2, 7, 10, 15] have also been studied on various kinds of models.

From the perspective of graph, one can transform a real traffic network into a graph in which vertices represent the intersections of roads, and edges represent the road segments. This idea was first proposed by Yang and Yeh [16], for which they also presented some properties. They also proved that the optimal setting for traffic lights is NP-hard. Later, Dong [4] considered the flow of vehicles on the graph model. Based on Dong's idea, in this paper, we first propose a new traffic

*This research work was partially supported by the National Science Council of Taiwan under NSC-95-2221-E-110-102.

model and its light setting problem in Section 2. Then, for solving the problem, a branch and bound strategy [12] and three evolutionary algorithms, the genetic algorithm (GA) [6, 13, 10], the particle swarm optimization (PSO) [6, 11] approach and the ant colony optimization (ACO) algorithm [6] will be described in Section 3. In Section 4, we show how to extend our model such that each car can turn its direction. Our experimental results and comparisons are written in Section 5. Finally, we draw our conclusions in Section 6.

2 The Traffic Graph Model and Its Properties

2.1 The Traffic Graph Model

Because the traffic network of a city is very complex in reality, to simplify the problem we also make some assumptions in our traffic model as follows.

1. Each traffic light has only two signals, red light and green light.
2. All traffic lights have the same cycle time T, the total duration of one red light and one green light.
3. The duration of the red light is equal to that of the green light.
4. Each traffic light is implemented with a two-phase signal, where some vehicles pass through the intersection during the green light and the other vehicles pass through the intersection during the red light (their direction is green light).
5. Each intersection has two traffic lights with opposite signals. Therefore, a single traffic light is sufficient to represent these two lights.
6. Each vehicle can either move forward or stay at its current position. That is to say, vehicles are not allowed to turn left or turn right.
7. If a vehicle reaches an intersection and the traffic light is red, it must wait until the traffic light turns green.
8. Each vehicle moves one unit of distance forward in one unit of time, provided that it is not blocked by a front vehicle or a red traffic light.

With the above assumptions, we can transform the traffic network of a city into a traffic graph model. Here we use a graph $G = (V, E)$ to represent the traffic network of a city, where each vertex $v \in V$ denotes the intersection of some roads and each edge $(u, v) \in E$ denotes the road segment between two vertices u and v. Since each intersection $v \in V$ is controlled by a traffic light, we use $t(v)$ to denote the starting time of green light on v. That is, the interval of green light on v can be written as $[t(v), (t(v) + T/2 - 1) \ mod \ T]$.

In a real traffic network, each intersection covers more than one road (usually two), where vehicles on one road must wait for the vehicles on the other road. In order to represent different phases in which the vehicles pass through the intersection, we associate each edge (u, v) with a pair of integers (i, j), where $i, j \in \{0, 1\}$. If $j = 1$, the vehicles from u to v can pass through v during the time interval $[t(v), (t(v) + T/2 - 1) \ mod \ T]$, while they must stop in front of v during the time interval $[(t(v)+T/2) \ mod \ T, t(v)-1]$. For the case of $j = 0$, the vehicles from u to v must stop in front of v during the time interval $[t(v), (t(v)+T/2-1) \ mod \ T]$, while they can pass through v during the time interval $[(t(v) + T/2) \ mod \ T, t(v) - 1]$. In other words, $t(v)$ denotes either the starting time of green light on v when $j = 1$, or the starting time of red light on v when $j = 0$.

In our model, each edge $(u, v) \in E$ is bidirectional. Let $d(u, v)$ denote the length of the road segment between u and v, where $d(u, v) = d(v, u)$. Note that $d(u, v)$ also denotes the time required for a vehicle moving from u to v. Because at most $T/2$ vehicles with the same direction can pass through the intersection v in one cycle time, to avoid the condition of traffic jam, we assume that $d(u, v) \geq T/2$ in our model, for each $(u, v) \in E$.

2.2 The Definition of Our Problem

In this section, we will introduce how we measure the fitness score of a given traffic light setting on the traffic graph model. When a vehicle waits for a green signal or other vehicles, a penalty is generated. For each intersection, we calculate its penalty by summing up the penalties generated by all waiting cars. After that, by summing up the penalties of all intersections, we obtain the total penalty of the given setting. The less total penalty is generated, the better the traffic light setting is.

An illustration for computing penalty is given in Figure 1, which shows two different traffic light settings. In Figure 1(a), we simulate the vehicle

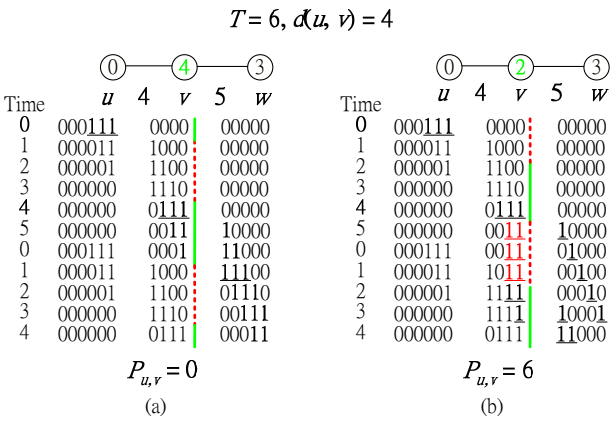

Figure 1: Penalty of vehicles in one direction on an edge. The solid line segments denote the green interval and the dash line segments denote the red interval.

moving on the path with the traffic light setting $t(u) = 0, t(v) = 4$ and $t(w) = 3$. It is simulated with a binary array, where 1 means there is a vehicle on the position and 0 means no vehicle. Each vehicle moves one unit of distance forward in each unit of time. So every car in the binary array moves to right, provided that it is not blocked by a front 1 or a red traffic light. In time 0, three vehicles move from the starting intersection u to v. Because $d(u, v) = 4$, these vehicles reach intersection v at time 4. Then, all vehicles pass through v one by one during the period of green light $[t(v), (t(v)+T/2) \ mod \ T] = [4, 0]$. Since there is no penalty produced by these vehicles on v, we have $P_{u,v} = 0$. In Figure 1(b), we set $t(v) = 2$. These vehicles also reach v at time 4. Because the period of the green light now becomes the internal $[2, 4]$, only the first vehicle can pass through v. At time 5, the traffic light turns red signal. The second vehicle waits for the next green signal, thus the third vehicle is also blocked by the second vehicle. These vehicles must wait until the traffic signal turns green, so $P_{u,v} = 2 \times T/2 = 6$. One can see that different settings cause different penalties.

In our traffic model, our goal is to find out how to set $t(u)$, $t(v)$ and $t(w)$ such that the total penalty is minimized. To end this section we give a more formal definition of the traffic light setting problem as follows.

Definition 1. *Given a global cycle time T, a traffic graph $G = (V, E)$ where each edge (u, v) is associated with a distance $d(u, v)$ and each starting intersection v is associated with a traffic flow $f(u, v)$, the traffic light setting problem is to find the optimal setting $t(v)$ for each $v \in V$ such that the total penalty $(\sum_{u,v \in V} P_{u,v})$ is minimized.*

3 Methodology

Since the setting on one intersection affects the traffic flow on other intersections, finding the optimal solution becomes very complicated. In this section, we first propose a branch and bound method to find out the optimal solution. In addition, we also propose three evolutionary algorithms, based on genetic algorithm (GA), particle swarm optimization (PSO) and ant colony optimization (ACO) to get the near optimal solutions.

3.1 Input Data

Our input is a traffic graph which is obtained by transforming from a traffic network. The detailed content is shown as follows.

1. A traffic cycle time T.
2. A distance matrix D, where each element $d_{u,v} \in D$ denotes the distance of each edge $(u, v) \in E$. Note that $d(u, v)$ is measured by time units required for moving from u to v.
3. A traffic flow matrix F, where $f_{u,v} = f(u, v)$ denotes the number of vehicles moving from u to v in one cycle.
4. A boolean color matrix B, where $b_{u,v} = b(u, v) \in \{0, 1\}$ denotes the corresponding color of traffic light at time $t(v)$ for vehicles moving from u to v.
5. The information of several edge-disjoint paths H on the traffic graph. In H, we assume that any two paths do not share a common edge and $\cup_{e \in H} e = E$.

3.2 The Branch and Bound Method

We begin with transforming the traffic light setting problem into a tree searching problem. Take Figure 2 as an example, any internal node in the tree has T children, and each light is represented as a single level. Therefore, any path from root to a leaf node denotes a feasible solution. In order to reduce the space complexity, we adopt the idea of *hill climbing* [12] scheme in our searching. Also, we use some tricks to make the branch and bound method work efficiently.

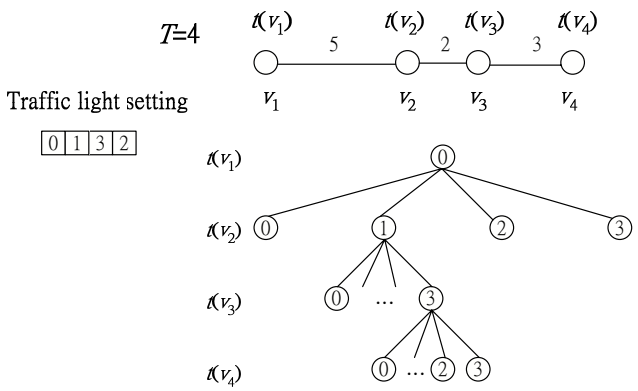

Figure 2: The tree representation of the traffic light setting problem.

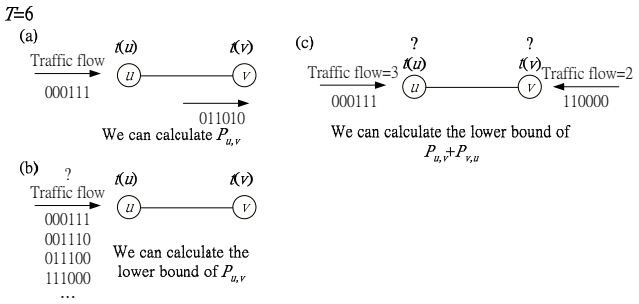

Figure 3: The traffic flow from u to v.

First, we propose a simple heuristic for traffic light setting, called *wave setting*. With the length of edge and current setting, wave setting sets the traffic light at the next intersection. The formula of wave setting is given as follows.

Wave setting :

$$t(v_i) = t(v_{i-1}) + d(v_{i-1}, v_i) \ mod \ T \qquad t(v_1) = 0 \tag{1}$$

In our branch and bound method, the solution of wave setting serves as the first upper bound of the optimal solution. It can be used to efficiently terminate futile branches. In our method, a node in the tree represents a state of the traffic light setting. If it is a leaf node, then present setting contains all intersections. Otherwise, the traffic light of some intersections have not yet been set. The lower bound of a node, which is obtained by summing up the penalties from the root to this node, denotes the lower bound of all possible solutions produced by this node. For an internal node, which means that some intersections are still to be set, we shall describe how to measure the lower bound of this node.

Figure 3 shows various conditions on two adjacent intersections u and v. If $t(u)$, $t(v)$ and the traffic flow from u to v are given, $P_{u,v}$ can be obtained by simulating the traffic flow from u to v. After that, we can also simulate the traffic flow from v to its next intersection. If the expanding node is already a leaf node, the settings of all intersections have been done. Therefore, the lower bound of this node is the penalty obtained by simulating traffic flows on all intersections, which therefore produces a solution. If the expanding node is not a leaf node, we obtain its lower bound by first getting the present penalty, which can be done by simulating each traffic flow $F_{u,v}$ whose $t(u)$ and $t(v)$ are already known, as shown in Figure 3(a). Next, for every road segment (u, v) with known $t(u)$, $t(v)$ and unknown $F_(u, v)$, we can still estimate the minimum $P_{u,v}$ by simulating all possible flows, as shown in Figure 3(b). Finally, in Figure 3(c), for two traffic flow $F_{x,u}$ and $F_{y,v}$ with unknown $t(u)$ and $t(v)$, we generate the set Z, where each element $z \in Z$ denotes a setting $t(u) = 0$ and $t(v) = z$ such that $P_{u,v} = 0$. Therefore, we can then determine the best z with the minimal penalty $P_{v,u}$ by setting $t(u) = 0$ and $t(v) = z$. That is, $P_{v,u}$ functions as a part of our lower bound. We regard the lower bound of each node as the cost of our searching.

3.3 The Genetic Algorithm

In our genetic algorithm, a chromosome represents a solution of traffic light setting on the traffic graph. Take the setting in Figure 2 as an example, the corresponding chromosome would be "0132". The fitness of each chromosome is measured by the total penalty of vehicles on this traffic light setting. The crossover is done by randomly selecting two chromosomes $A = a_1a2\cdots a_N$ and $B = b_1b_2\cdots b_N$ from the population. Then, via exchanging information, A and B produce a new offspring $C = c_1c_2\cdots c_N$. We choose a value k, where k is the half length of chromosome plus a random value p. In our method, p falls within the range from -2 to 2. If index $i \leq k$, gene $c_i = a_i$. Otherwise, gene $c_i = b_i$. The mutation is done by randomly changing genes on the offspring C, with the mutation probability.

3.4 The Particle Swarm Optimization Method

Let $Y_i = y_{i_1}, y_{i_2}, \cdots, y_{i_N}$ be a setting in the model. In our PSO, to represent a bird with

Y_i, the corresponding position of Y_i is encoded as $X_i = x_{i_1}, x_{i_2}, \cdots, x_{i_{N-1}}$, where $x_{i_j} = y_{i_{j+1}} - y_{i_j}$, for $1 \leq j \leq N-1$. This kind of coding is called $\Delta-$*encoding*. The fitness of each particle is also evaluated by its total penalty on the traffic network. During the flight, each particle p_i keeps three records as follows:

$$\begin{aligned} &current\ setting\ X_i = (x_{i_1}, x_{i_2}, \cdots, x_{i_{N-1}}) \\ &previous\ best\ B_i = (b_{i_1}, b_{i_2}, \cdots, b_{i_{N-1}}) \\ &flying\ velocity\ V_i = (v_{i_1}, v_{i_2}, \cdots, v_{i_{N-1}}) \end{aligned}$$

In each flying cycle, the traffic light setting B_g of the best particle g is calculated as the best fitness of all particles. Each particle updates its velocity V_i as follows:

Velocity update :

$$\begin{aligned} new\ V_i = \omega \times current\ V_i + c_1 \times rand() \times (B_i - X_i) \\ + c_2 \times rand() \times (B_g - X_i) \end{aligned} \tag{2}$$

The parameter ω controls the impact of previous velocity, c_1 and c_2 denote two learning factors. $rand()$ is the random function generating random number in the range $[0, 1]$. Using the new velocity V_i, the particle p_i updates its solution as follows:

Traffic light setting update :

$$\begin{aligned} X_i &= X_i + new\ V_i \\ V_{max} &\geq V_i \geq -V_{max} \end{aligned} \tag{3}$$

The parameter V_{max} denotes the maximum change of a particle velocity.

3.5 The Ant Colony Optimization Algorithm

In our ACO, the route of a ant represents a solution on the traffic graph, which is encoded with $\Delta-$*encoding*. For example, if we have $|V| = 4$ and $V = \{v_1, v_2, v_3, v_4\}$ on the traffic graph, the route of each ant consists of three vertices l_1, l_2 and l_3, where l_i denotes $t(v_{i+1}) - t(v_i)$. The formula of probability can be rewritten as follows:

Probability :

$$pr_k(i,j) = \frac{[\tau(i,j)(t)]^\alpha [\eta(i,j)]^\beta}{\sum_{u=0}^{T-1} [\tau(i,u)(t)]^\alpha [\eta(i,u)]^\beta}. \tag{4}$$

Here, $pr_k(i,j)$ denotes the probability that the kth ant selects the jth outgoing edge of the vertex l_i, $0 \leq j \leq T-1$. $\tau(i,j)(t)$ denotes in iteration t the pheromone concentration on the jth outgoing edge of l_i. Also, $\eta(i,j)$ represents the weight of the jth outgoing edge of l_i, which can be obtained as

$$\frac{1}{\frac{2*n}{T} * Penalty(v_i, v_{i+1}) + \frac{2*m}{T} * Penalty(v_{i+1}, v_i)}. \tag{5}$$

In Equation 5, $Penalty(v_i, v_{i+1})$ means the penalty obtained by simulating $T/2$ vehicles from v_i to v_{i+1} with $t(v_i) = 0$ and $t(v_{i+1}) = j$. The variables n and m represent traffic flows from v_i to v_{i+1} and v_{i+1} to v_i, respectively. α and β are two parameters that control the influence of the pheromone trails and the weight. The pheromone concentration parameter $\tau(i,j)(t)$ will be updated as follows:

Pheromone update :

$$\tau(i,j)(t+1) = (1-\rho) \times \tau(i,j)(t) + \sum_{k=1}^{q} \triangle\tau(i,j)^k(t). \tag{6}$$

In Equation 6, the parameter ρ is the pheromone evaporation rate, where $0 < \rho < 1$. q denotes the total number of ants and $\triangle\tau(i,j)^k(t)$ is the pheromone left by the kth ant on the jth outgoing edge of l_i. If the kth ant selects the jth outgoing edge of l_i, we set $\triangle\tau(i,j)^k(t) = \frac{10}{(\sum_{u,v \in V} P_{u,v})^k}$, where $(\sum_{u,v \in V} P_{u,v})^k$ denotes the total penalty caused by the setting of the kth ant. Otherwise, we set $\triangle\tau(i,j)^k(t) = 0$.

4 The Extended Traffic Graph Model

One of the assumptions on the traffic graph model is that vehicles are not allowed to turn left or right at an intersection. This is an unrealistic rule for the real vehicle movement. In this section, we try to consider the situation that a vehicle may turn left or right at an intersection. In this extended traffic graph model, when a vehicle reaches an intersection and the corresponding traffic light is green (which allows it to turn), it may turn left or right with a probability. Because there may be two opposite vehicles which attempt to enter the same road, we assume that the vehicle turning right has the higher priority than the vehicle turning left. The condition of traffic jam may happen when too many vehicles enter the same road.

Since a vehicle usually slows down when it turns its direction. There should be a penalty when a vehicle turns left or right. Actually, whenever a vehicle cannot move with its normal speed, there should be a penalty. Formally, the penalty is generated with the following rules. (1)The penalty

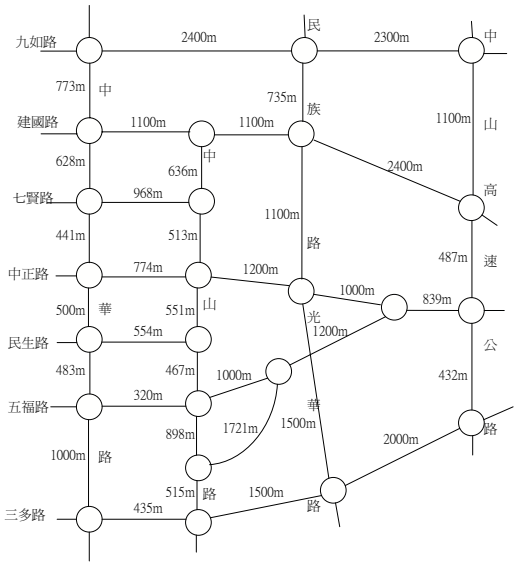

Figure 4: A simple traffic graph transformed from the map of Kaohsiung city in Taiwan.

P_m for any vehicle that stops and waits for a red signal to become to green. (2)The penalty P_n for any vehicle which stops and waits for other vehicles. (3)The penalty P_l for any vehicle that turns left. (4)The penalty P_r for any vehicle that turns right. (5)The penalty P_s for any vehicle that reaches an intersection and slows down for the red light.

Because each vehicle turns with certain probability, the result of each simulation may be different. Hence, there is no optimal solution on the extended model. We only calculate one estimated solution to represent the best solution on the extended model. If the turning probability is zero on all intersections, the extended traffic graph model becomes the original traffic graph model.

5 Experimental Results

First, we test the simplest cases with $|V| = 4$ for $T = 10$, $T = 30$ and $T = 60$. Then, we test the general cases with $|V| = 9$ for $T = 10$ and $T = 30$. Furthermore, we transform the partial map of Kaohsiung city in Taiwan into a traffic graph as shown in Figure 4. Our testing environment is a PC with AMD Athlon XP 1800+ CPU and Windows XP SP2 operating system. Our compiler is Visual Studio .Net C++ 2003. In our experiment, each case is tested with each algorithm for 10 times.

The parameters of the penalty on the original traffic graph model are set with $P_m = 1$ and $P_n = 1$. The parameters of the penalty on the extended traffic graph model are set with $P_m = 1$, $P_n = 1$, $P_l = 0.6$, $P_r = 0.5$ and $P_s = 0.7$. These setting of parameters are obtained via several real person observations on some intersections of Kaohsiung city. Furthermore, the probabilities of turning left or right are both set to 0.1 for all vehicles on all intersections. The parameters of our GA, PSO and ACO on the original traffic model and that of the extended model are shown in Table 1.

The experimental results for all test cases on the traffic graph model are shown in Table 2. In this table, $N(V)$ denotes the number of vehicles entering the traffic graph at all starting intersections in one cycle time, and the term "Penalty" means the total penalty resulted from per vehicle on each intersection in one cycle time divided by the cycle time T. In other words, this penalty represents the average waiting time in percentage of T. Because the length of each road segment is greater than or equal to $T/2$, each vehicle waits at most $T/2$ time before it passes through the intersection. Therefore, the upper bound of the penalty is 0.5.

For test cases 1, 2 and 3, we can see that all of GA, PSO and ACO efficiently obtain the optimal solutions since the given graphs are small. For test case 4, GA and PSO still find the optimal solution efficiently, while the required time of the branch and bound algorithm increases rapidly with $|V|$ and $|T|$. Then, the result of test case 5 indicates that the branch and bound algorithm is quite time consuming, though the optimal solution can still be obtained. At the same time, GA, PSO and ACO are still able to find near optimal solutions in a short period of time. Therefore, we only adopt GA, PSO and ACO to find near optimal solutions of test cases 6 and 7. In fact, GA can even report the optimal solution with its execution time less than half of that in PSO and ACO.

The experimental results on the extended traffic graph model are shown in Table 3. Because test case 7 is quite time consuming, we do not test the entire traffic graph. For the smaller test cases 1, 2, 3 and 4, we can see that the penalty of GA, PSO and ACO are similar. For the larger test cases 5 and 6, the results indicate that the average, minimum and maximum penalties of GA are a little better than those of PSO and ACO. The execution time of GA is about half of that in PSO and ACO. Take the execution time and total penalty into account, GA achieves the best results.

Table 1: The parameters of our GA, PSO and ACO on the traffic graph and the extended traffic graph models. The dashes denote that the algorithm does not have the parameter.

Model	The traffic graph model						The extended traffic graph model					
Cases	$\|V\| = 4$			$\|V\| = 9$, $\|V\| = 24$			$\|V\| = 4$			$\|V\| = 9$, $\|V\| = 24$		
Algorithm	GA	PSO	ACO	GA	PSO	ACO	GA	PSO	ACO	GA	PSO	ACO
Population size	100	100	100	1000	1000	1000	50	50	50	100	100	100
Number of generations	500	500	500	500	500	500	100	100	100	200	200	200
Crossover rate	0.5	-	-	0.5	-	-	0.5	-	-	0.5	-	-
Mutation rate	0.03	-	-	0.03	-	-	0.03	-	-	0.03	-	-
The velocity V_{max}	-	$T/5$	-	-	$T/5$	-	-	$T/5$	-	-	$T/5$	-
The inertia weight w	-	1	-	-	1	-	-	1	-	-	1	-
The weight α	-	-	1	-	-	1	-	-	1	-	-	1
The weight β	-	-	1	-	-	1	-	-	1	-	-	1
The weight ρ	-	-	0.7	-	-	0.7	-	-	0.7	-	-	0.7

Table 2: The experimental results of our four algorithms on the traffic graph model, where BB denotes the branch and bound strategy, and $N(V)$ denotes the number of vehicles.

Case	Parameter	Algorithms	Avg. Penalty	Min. Penalty	Max. Penalty	Avg. Time(s)
1	$\|V\| = 4, T = 10,$ $N(V) = 28$	BB	optimal solution = 0.100			0.2
		GA				1.7
		PSO				2.1
		ACO				2.3
2	$\|V\| = 4, T = 30,$ $N(V) = 110$	BB	optimal solution = 0.160			0.6
		GA				2.1
		PSO				2.6
		ACO				3.0
3	$\|V\| = 4, T = 60,$ $N(V) = 220$	BB	optimal solution = 0.0901			1.0
		GA				2.5
		PSO				3.4
		ACO				3.6
4	$\|V\| = 9, T = 10,$ $N(V) = 42$	BB	0.0904	0.0904	0.0904	101.9
		GA	0.0904	0.0904	0.0904	27.9
		PSO	0.0904	0.0904	0.0904	37.7
		ACO	0.0928	0.0928	0.0928	48.1
5	$\|V\| = 9, T = 30,$ $N(V) = 134$	BB	0.0924	0.0924	0.0924	≥ 21600
		GA	0.0946	0.0924	0.1008	36.4
		PSO	0.0945	0.0924	0.1004	51.7
		ACO	0.0984	0.0965	0.1008	62.5
6	$\|V\| = 24, T = 80,$ $N(V) = 52$	GA	0.0625	0.0602	0.0669	113.3
		PSO	0.0773	0.0692	0.0853	245.3
		ACO	0.1166	0.1069	0.1229	244.4
7	$\|V\| = 24, T = 120,$ $N(V) = 181$	GA	0.0650	0.0623	0.0672	136.3
		PSO	0.0770	0.0703	0.0826	292.4
		ACO	0.1215	0.1168	0.1287	304.3

Table 3: The experimental results of our GA, PSO and ACO on the extended traffic graph model.

Case	Parameter	Algorithms	Avg. Penalty	Min. Penalty	Max. Penalty	Avg. Time(s)
1	$\|V\| = 4, T = 10,$ $N(V) = 28$	GA	0.1044	0.0950	0.1094	8.2
		PSO	0.1013	0.0937	0.1096	15.4
		ACO	0.1084	0.0910	0.1254	15.5
2	$\|V\| = 4, T = 30,$ $N(V) = 110$	GA	0.1500	0.1399	0.1579	36.9
		PSO	0.1494	0.1433	0.1555	71.8
		ACO	0.1592	0.1516	0.1720	70.7
3	$\|V\| = 4, T = 60,$ $N(V) = 220$	GA	0.1410	0.1336	0.1583	93.3
		PSO	0.1398	0.1312	0.1523	184.4
		ACO	0.1347	0.1221	0.1480	183.8
4	$\|V\| = 9, T = 10,$ $N(V) = 42$	GA	0.1115	0.0941	0.1254	171.6
		PSO	0.1218	0.1044	0.1344	332.4
		ACO	0.1130	0.1033	0.1254	333.6
5	$\|V\| = 9, T = 30,$ $N(V) = 134$	GA	0.1348	0.1207	0.1472	774.9
		PSO	0.1370	0.1308	0.1501	1509.6
		ACO	0.1439	0.1316	0.1571	1563.2
6	$\|V\| = 24, T = 80,$ $N(V) = 52$	GA	0.1306	0.1254	0.1398	31292.3
		PSO	0.1411	0.1339	0.1520	62893.9
		ACO	0.1588	0.1497	0.1656	63481.2

6 Conclusion

In this paper, we adopt the traffic graph model to represent the traffic network, then propose four algorithms to minimize the total waiting time of each vehicles. First, we propose the branch and bound method which obtains the optimal solution. However, it is quite time consuming for large graphs with long cycle time T. Of three evolutionary algorithms GA, PSO and ACO, our experiments show that GA seems to be an efficient approach which fits our problem. In our extended model, we make the assumption that each vehicle may change its direction. The experimental results again reveal that GA is a good strategy to obtain better solutions in our graph model. In our experiment, we also test our algorithms on the graph obtained from the traffic data of Kaohsiung city. In the future, we shall try to study for useful properties in our model which might be used to speed up the branch and bound algorithm. In addition, the solution yielded by GA might also be a better bound for cutting hopeless branches.

References

[1] O. Biham, A. Middleton, and D. Levine, "Self-organization and a dynamical transition in traffic-flow models," *Physical Review A*, Vol. 46, No. 10, pp. R6124–R6127, Nov. 1992.

[2] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg, "Optimizing traffic lights in a cellular automaton model for city traffic," *Physical Review E*, Vol. 64, No. 5, p. 056132, Oct. 2001.

[3] D. Chowdhury and A. Schadschneider, "Self-organization of traffic jams in cities: Effects of stochastic dynamics and signal periods," *Physical Review E*, Vol. 59, No. 2, pp. R1311–R1314, Feb. 1999.

[4] J.-F. Dong, *Optimizing the Traffic Signal Setting Problem on the Graph Model*. Master Thesis, National Sun Yat-Sen University, Kaohsiung, Taiwan, July 2006.

[5] R. M. D'Souza, "Geometric structure of coexisting phases found in the biham-middleton-levine traffic model," *Physical Review E*, Vol. 71, 2005.

[6] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced Engineering Informatics*, Vol. 19, pp. 43–53, 2005.

[7] M. E. Fouladvand and M. Nematollahi, "Optimization of green-times at an isolated urban crossroads," *The European Physical Journal B*, Vol. 22, pp. 395–401, 2001.

[8] M. E. Fouladvand and N. H. Radja, "Optimized traffic flow at a single urban crossroads: dynamical symmetry breaking," tech. rep., Institute for Studies in Theoretical Physics and Mathematics, 2001.

[9] M. E. Fouladvand, Z. Sadjadi, and M. R. Shaebani, "Optimized traffic flow at a single intersection: traffic responsive signalization," *Journal of Physical A: Mathematical and General*, Vol. 37, pp. 561–576, 2004.

[10] T. Kalganova, G.Russell, and A. Cumming, "Multiple traffic signal control using a genetic algorithm," *Proc. of the Fourth International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'99).*, Portoroz, Slovenia, pp. 220–228, 1999.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of the IEEE international conference on neural networks*, Vol. 4, Perth, Australia, pp. 1942–1948, 1995.

[12] R. C. T. Lee, S. S. Tseng, R. C. Chang, and Y. T. Tsai, *Introduction to the Design and Analysis of Algorithms*. McGraw-Hill, 2005.

[13] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, London, UK, 1996.

[14] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," *Journal of Physical I France*, Vol. 2, pp. 2221–2229, 1992.

[15] I. Porche, M. Sampath, R. Sengupta, Y.-L. Chen, and S. Lafortune, "A decentralized scheme for real-time optimization of traffic signals," *Proceedings of the 1996 IEEE International Conference on Control Applications*, Dearborn, MI, pp. 582–589, Sept. 1996.

[16] C. B. Yang and Y. J. Yeh, "The model and properties of the traffic light problem," *Proc. of International Conference on Algorithms*, Kaohsiung, Taiwan, pp. 19–26, Dec. 1996.