For office use only
T1 ________________
T2 ________________
T3 ________________
T4 ________________

Team Control Number
67325

Problem Chosen
C

For office use only
F1 ________________
F2 ________________
F3 ________________
F4 ________________

2017
MCM/ICM
Summary Sheet

(Your team's summary should be included as the first page of your electronic submission.)

Type a summary of your results on this page. Do not include the name of your school, advisor, or team members on this page.

Summary

In order to find out the connection among percentage of self-driving cars, traffic flow, traffic density, speed and number of lanes, we propose a model based on cellular automaton. Then the model is applied to the data for the roads of interest.

First, we propose our model based on cellular automaton from the microscopic view. We first study Single Lane, and the car can't communicate with each other. According to characteristics of different kinds of cars, we introduce the principle of Safety Distance to the model. We define the biggest difference between two kinds of cars is the different reaction time. Then we study Multiple Lanes. We introduce Lane Changing step to the model. When actual speed is less than what driver hopes, driver comes up with conscious of lane changing. The car changes to another lane with probability γ if requirement of speed and distance is coincident. After that, we introduce cooperating car to the model. The motion state identical for adjacent cars is the most important characteristic of self-driving cars. So the distance could almost be ignored. The motion state of car is changed according to the front car, and their state changes at the same time. The above are our main steps for modeling, and we use C++ program to implement our model.

Then, we apply our model to study the effects on traffic flow of the number of lanes, average traffic volume, and percentage of vehicles using self-driving, cooperating systems. After assuming some data of cars and roads, we get traffic flow from the model while percentage, initial traffic flow and number of lanes change. In order to observe conveniently, we provide some relation curves to find out the influence to traffic flow from changing of percentage.

Next, we apply our model to study the effects on traffic flow during peak travel hours. We work out the peak traffic time of different percentages of self-driving cars and the different roads. We put forward the appropriate percentage of self-driving cars to ease traffic congestion effectively by analyzing equilibrium during peak traffic hours. The result is helpful for government changing policy.

Last, we analyze the sensitivity and stability of the model. We give tiny disturbance to the percentage, and analyze the change of time of passage during peak traffic hours.

A Letter to the Governor's office

Dear Governor,

We propose our model based on cellular automaton from the microscopic view. After comparing with actual traffic situation, we introduce rules of Single Lane, Multiple Lanes and Cooperating to our model. And considering cooperation between self-driving cars as well as the interaction between self-driving and non-self-driving vehicles, the mixed traffic model comes into being.

Then according to limitations of our model, the data and map, we find out the connection among several variables to solve the problem accurately.

Because self-driving cars could communicate with each other, the adjacent vehicles have the same motion state. And the distance between adjacent self-driving cars could be ignored. But there is still a safety distance between self-driving cars and the car in the existing traffic flow.

Normally, the traffic flow increases with the increase of percentage of self-driving cars while the number of lanes is a constant, and the tipping point exists. Meanwhile, the passage time of each section is influenced by the percentage during peak traffic hours. The passage time becomes short with the increase of percentage, which is helpful to ease traffic jam. There is an equilibria point in the graph of the connection between percentage and passage time. The percentage of equilibria point means the best one to ease traffic jam. Lanes are dedicated to these cars when equilibria point=100%.

According to the results and other factors that may affect the accuracy of our model, the advice we give to the government is :

1. The proportion of self-driving cars that government should put into on Interstates 5, 90, and 405, as well as State Route 520, ranges from 50%~60%.
2. The section of 405B can be dedicated to these self-driving cars.
3. If possible, the government can increase the investment in self-driving cars.
4. Reduce the cost of self-driving cars' production.
5. Increase the size or capacity of self-driving cars, and expand them.

In a word, our model produces the equilibrium point between the peak traffic time and the proportion of self-driving cars. And then take the equilibria as the best proportion of self-driving car inputs. (The list of each section's equilibria location is attached at the end of the letter)

We have a strong belief that we are able to solve the problem of peak congestion in mixed traffic by using our model. And naturally, the optimal proportion of self-driving car inputs will be determined.

Route_ID (New)	Route_ID	Start comment	End comment	equilibria
5A	5	Olympia	Rte 101 intersection	0
5B	5	Rte 101 intersection	SR 510 intersection	0.5
5C	5	SR 510 intersection	SR 512 intersection	0.55
5D	5	SR 512 intersection	SR 16 Intersection	0.52
5E	5	SR 16 Intersection	I705 intersection	0.53
5F	5	I705 intersection	SR167 Joins	0.58
5G	5	SR167 Joins	I405 intersection	0.49
5H	5	I405 intersection	I90 intersection	0.52
5I	5	I90 intersection	SR 520 intersection	0.52
5J	5	SR 520 intersection	I405 intersection	0.54
5K	5	I405 intersection	Snohomish County	0.48
90A	90	Downtown Seattle	Intersection with I5	0.15
90B	90	Intersection with I5	Intersection with I405	0.55
90C	90	Intersection with I405	/	0.4
405A	405	/	/	0.54
405B	405	/	/	0.58
405C	405	/	/	0.53
405D	405	/	/	0.5
520A	520	/	/	0.46
520B	520	/	/	0.52

Yours Sincerely,
Team #67325

Contents

1 Introduction

1.1 Statement of the Problem

In the United States, due to the number of lanes of roads, the traffic capacity is limited. In the Greater Seattle area, particularly on Interstates 5, 90, 405, and State Route 520, the problem of vehicles is especially obvious during the rush hour. Practice shows that the introduction of unmanned vehicle can improve the highway traffic lane capacity without increasing the number of lanes. The following is what we need to solve in the modeling problem.

(1) What is the cooperation between self-driving cars as well as the interaction between self-driving and non-self-driving vehicles?

(2) In the case of a certain number of lanes, peak and average traffic volume, and (1) in the presence of interaction, when changing the proportion of self-driving vehicles (from 10% to 50% to 90%), how does the road traffic situation respond to change?

(3) Is there any equilibrium between the road traffic situation and the proportion of self-driving vehicles?

(4) Is there a tipping point where performance changes markedly, and under what circumstances does it need to provide dedicated lanes for self-driving cars?

(5) Does the model also reflect the impact of other policies change on traffic capacity?

Note: the road distribution used in the above model is got from the map given, and the data are got from the sections of the Excel, intersections and traffic flow data.

1.2 Assumptions & Detailed Definitions

1.2.1 Assumptions

A1.Each car’s length is identical.

A2.Each car’s actual speed limit for all these roads is 60 miles per hour.

A3.A traffic jam is mainly caused by the probability of slowing down and slow response.

A4.The characteristic of self-driving cars is the short reaction time.

A5.The characteristic of cooperation is the same motion state by sharing information.

.

1.2.2 Detailed Definitions

Definitions used in our model:

Name	Definition	Denotation	Unit
Traffic flow	The number of vehicles per unit time.	Q	M/h
Traffic density	The number of vehicles per unit distance.	k	M/km
Self-driving car percentage	The percentage of self-driving car of all vehicles.	P	/
Vehicle spacing	The distance between the head of vehicle n and the tail of the vehicle in front of vehicle n.	Gap_n	m
Randomization probability	The probability of slowing down caused by human error.	R_P	/
The expected headway	The distance between before and after the car.	H_d	m
Location variables	Car(n)'s position at the moment of t.	$X_n(t)$	m
Speed variables	Car(n)'s speed at the moment of t.	$V_n(t)$	m/s
Maximum speed	The nominal speed limit for all these roads.	V_{max}	m/s

1.3 The Advantages of Our Model

1.The new hybrid traffic flow model based on safety distance can be used to reflect the evolution characteristics of hybrid traffic system more realistically
2.Our model considers the situation of multiple lanes, so it has a wider application.
3.Our model considers the cooperation and interaction between self-driving cars, and emphasizes the important affection of percentage to the Traffic capacity.
4.Our data is based on the accurate simulation, which is more reliable and convincing.

2 Modeling

2.1Model Overview

From the microscopic view, based on cellular automata, we adopt the principle of step by step, which means that our research develops from simple to numerous. And the effects on traffic flow including the number of lanes, peak and average traffic volume, and percentage of vehicles using self-driving, cooperating systems, are studied from single lane to multiple lanes, from no communication to have communication. We try to find out specific contact between the independent variables and several dependent variables including traffic flow, traffic density and speed. The core of our model is to come up with an answer of cars interacting with the existing traffic flow and each other.

2.2Single Lane

We first choose single lane as our object of study, and ignore the communication between self-driving cars as well as vehicle lane changing. The simplified method is conducive to our study. The research achievement could be fundamental of our deeper study.

Considering physical characteristics, speed, deceleration performance of self-driving and no self-driving cars, especially for the influence of response time to driving behavior, we introduce $Gap_{safe,n}$ to cellular automata:

$$Gap_n = x_{n+1}(t) - x_n(t) - \mathrm{l}_{n+1}$$
$$Gap_{safe,n} = v_n(t)\tau_n + \frac{v_n(t)^2}{2b_n} - \frac{{v_{n+1}}^2(t)}{2b_n}$$

$Gap_{safe,n}$ is a safe distance needed for the N^{th} car; $x_n(t)$ is the location where the N^{th} car is when the time t , and $v_n(t)$ is its speed at the same time; $(N+1)^{th}$ is in $x_{n+1}(t)$ in front of $x_n(t)$, whose body length is l_{n+1}, and $v_{n+1}(t)$ is its speed at the same time ; b_n is the maximum deceleration for the N^{th} car; τ_n is the driver's reaction time of the N^{th} car.

We mainly consider the difference between self-driving and no self-driving cars from the view of the reaction time. They are the same in the model but τ_n. Vehicles perform the corresponding evolution rules of the speed and position by comparing Gap_n and $Gap_{safe,n}$.

(1) Based on the safety distance principle, the N^{th} car's driver estimates the maximum deceleration of the vehicle ahead to ensure the safe distance $Gap_{safe,n}$ to their front and the safe speed $v_{safe,n}$:

$$
\begin{aligned}
& Gap_n = Gap_{safe,n} \\
& \Rightarrow x_{n+1}(t) - x_n(t) - l_{n+1} = v_n(t)\tau_n + \frac{v_n(t)^2}{2b_n} - \frac{v_{n+1}{}^2(t)}{2b_n} \\
& \Rightarrow v_{safe,n}(t) = -b_n\tau_n + \sqrt{b_n{}^2\tau_n{}^2 + b_n\{2[x_{n+1}(t) - x_n(t) - l_{n+1}] + \frac{v_{n+1}{}^2(t)}{b_n}\}}
\end{aligned}
$$

(2) Accelerate the rules

When $Gap_n > Gap_{safe,n}$, the vehicle is carried out in accordance with the following rules to accelerate driving to meet the driver for higher motion of the expected speed:

$$v_n(t) \to \min(v_n(t) + a_n, V_{\max}, v_{safe,n}, Gap_n)$$

(3) Uniform rules

When $Gap_n = Gap_{safe,n}$, in the case of ensure the safety of vehicle driving, it is Vehicles will not take any measures acceleration and deceleration, maintain the original speed:

$$v_n(t) \to \min(v_n(t), Gap_n)$$

(4) Slow down the rules

When $Gap_n < Gap_{safe,n}$, in order to ensure safe driving is to slow down. If $v_{n+1}(t)$=0, the rules of Security to slow down are chosen; if $v_{n+1}(t) \neq 0$, the rules of Certainty to slow down are chosen:

Security to slow down:

$$v_n(t) \to \max\{\min(v_{safe,n}(t), Gap_n - 1), 0\}$$

Certainty to slow down:

$$v_n(t) \to \max\{\min(v_{safe,n}(t), Gap_n), 0\}$$

(5) Randomization probability

Considering the existed uncertainty in the course of the drivers on the road driving , random probability of slowing down R_p is introduced in the rules:

$$v_n(t) \to \max(v_n(t) - b_n, 0)$$

(6) Location Update

Based on the velocity update rules ,update the location of vehicles:

$$x_n(t) \to x_n(t) + v_n(t)$$

2.3Multiple Lanes

When we consider multiple lanes, the most important part is not the increase in the number of lanes, but the cars change their lanes from one to another.

We assume traffic system is made by three parallel direction lanes. Every lane is regarded as 1 d discrete lattice chains and their length is L. Each grid point is empty or only dominated by a car in the lattice chain. Then we assume the maximum speed of the car in every lane is the same. The car in the lane move from left to right. The speed of cars will distribute from 0 to $V_{\max}$ .Every step of evolution is divided as two steps including traveling and lane changing during the evolution process of the model.

The traveling step is the same as Single Lane, so we will play an important role in Lane Changing step, and our model come up with some rules to imitate lane changing in the reality.

Lane Changing step (taking three lanes as an example):

Vehicles will change their lanes according to the needs of driving. The principles of overtaking and safety should be met during the lane changing process.

'i' is the serial number of current lane, 'j ' is the serial number of adjacent lanes; 'place' is the location in lane j where the car in lane i wants to be.

$Gap_k^{(i)}$ is the number of spaces in front of the car i in the lane k; V_{hope} is the value of expectation speed for drivers; $fGap_{j,i}$ is the number of spaces in front of the 'place'; $bGap_{j,i}$ is the number of spaces behind of the 'place'; $bV_{j,i}$ is the speed of the nearest car behind of the'place'; $\gamma_{i,j}$ is the possibility that the car in the i lane changes to the j lane.

(1) The vehicle lane changing rules for the first lane:

When $Gap_k^{(i)} < V_{hope}$, lane changing consciousness is created out. If $Gap_1^{(i)} <$ $fGap_{2,1}$

and $bGap_{2,1} \geq bV_{2,1}$, the first lane current vehicles with probability $\gamma_{1,2}$ turn to the second lane with speed remaining constant.

(2) In the second lane of the vehicle can be left to the first lane may also turn right to the third lane, lane changing rules are as follows:
(a)Rules of turning left:

When $Gap_2^{(i)} < V_{hope}$, lane changing consciousness is created out. If $Gap_2^{(i)} <$ $fGap_{1,2}$ and $bGap_{i,j} \geq bV_{i,j}$, the second lane current vehicles with probability $\gamma_{2,1}$ turn to the first lane with speed remaining constant.
(b)Rules of turning right:

If $Gap_2^{(i)} \geq fGap_{1,2}$ and $fGap_{3,2} >$ $Gap_2^{(i)}$ and $bGap_{3,2} \geq bV_{3,2}$, the second lane current vehicles with probability $\gamma_{2,3}$ turn to the third lane with speed remaining constant.

(3)The vehicle lane changing rules for the third lane:

When $Gap_3^{(i)} < V_{hope}$, lane changing consciousness is created out. If $Gap_3^{(i)} <$ $fGap_{2,3}$ and $bGap_{2,3} \geq bV_{2,3}$, the first lane current vehicles with probability $\gamma_{3,2}$ turn to the second lane with speed remaining constant.

2.4Cooperating Car Model (V2V)

A cooperating car communicates and exchanges data with other cooperating self-driving cars as it decides what to do, which means cars communicating with each other. The characteristic of cooperation for the car will change the model from single lane to multiple lanes. In order to describe cooperating car, we introduce the concept of V2V to the model.

V2V, a communication technology considering interconnection between cars, means vehicle-to-vehicle. According to characteristics of V2V technology, introduce parameter α to characterize the driver's degree of reaction in advance after receiving real-time traffic information provided by the V2V technology.

When cars aren't equipped with V2V device, self-driving cars control themselves in according to the change of the distance between vehicles: When $\triangle x_n$ becomes longer, speed up; otherwise, slow down. After introducing V2V to vehicles, self-driving cars could make operation to accelerate or decelerate when obtaining in all kinds of traffic state changes of the front vehicles accurately and in real time. At the moment, self-driving cars equivalent to the change of traffic flow, and react in the same time. With the development of V2V communication technology, self-driving cars could change the state of cars without delay when α =1; in addition, if α >1, there is another condition where self-driving cars could predict the running of traffic flow in the future based on the rules of traffic state changing, and change the running state of cars in advance.

In our model, we assume α =1, so self-driving cars could change the state of cars without delay. The distance between these cars adjacent could be infinitely short. Because the car always studies the motion state of the front car, so their motion state is exactly same, we could regard them as one car but longer. The rule only applies to adjacent cars, in other words, if there is a no self-driving car between them, their states are different.

When cars change their lanes, there is another problem couldn't be ignored. If the car changes to new lane, even if the car in front of it is self-driving, their motion state is different. The car has to spend some time changing speed to be the same motion state with the front car. Otherwise, the speed transient will appear.

2.5 the Program flow chart of the model

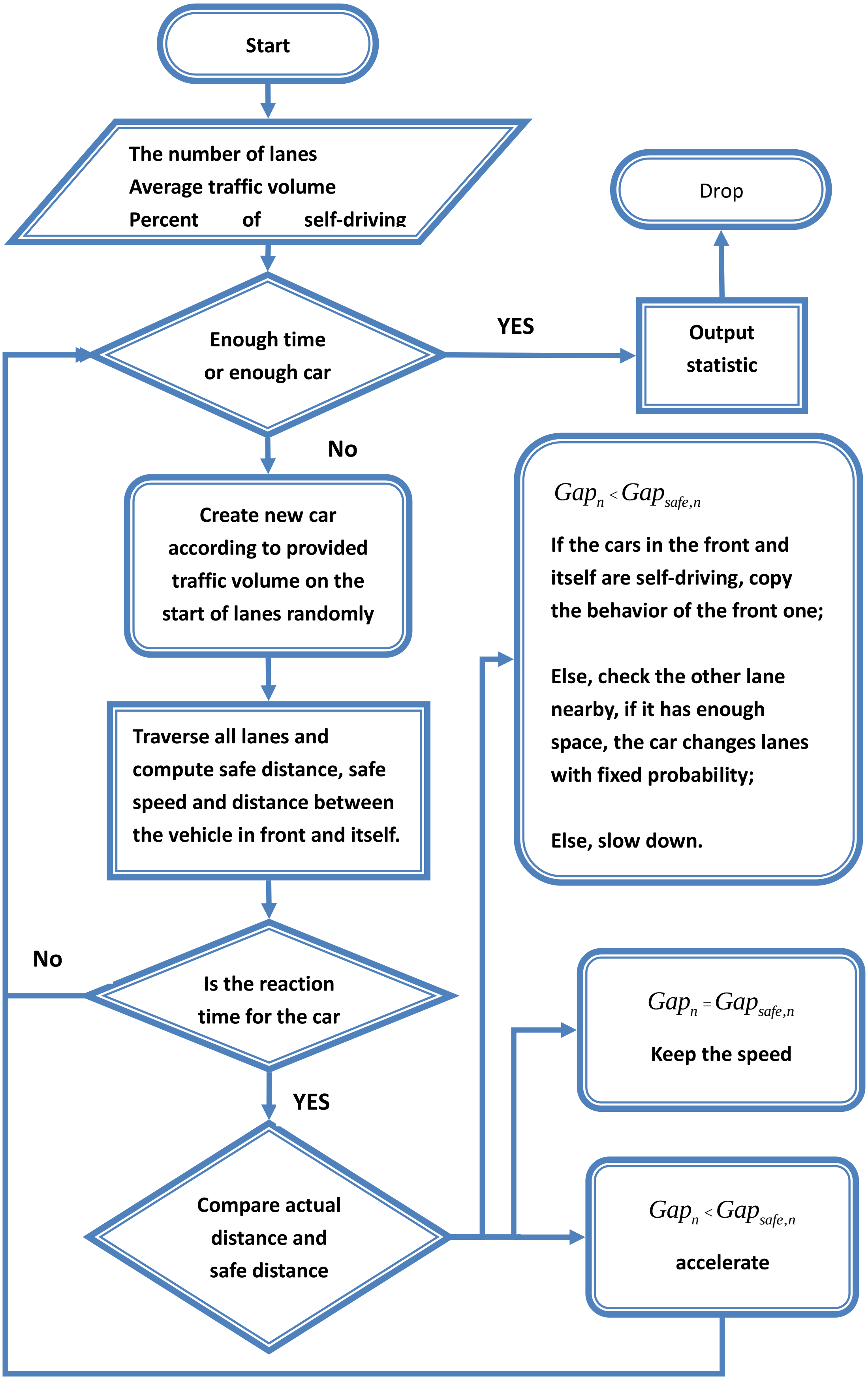


Fig.1 Program flow chart

3 Model Analysis

3.1Simulation Design

We assume there is a one direction road of three lanes 1.2km in length. Every cellular is 0.3m in length and every car occupies 12 cellular; $V_{\max}$ is 90 cellular/s(60miles/h), a_n is 10 cellular/ s^2, b_n is 10 cellular/ s^2; Unit time interval is 0.1s; randomization probability R_p is 0.4; lane changing probability γ_{ij} is 0.5; the response time of self-driving can is 1 unit time, the response time of no self-driving can is 10 unit time.

3.2Connection Analysis

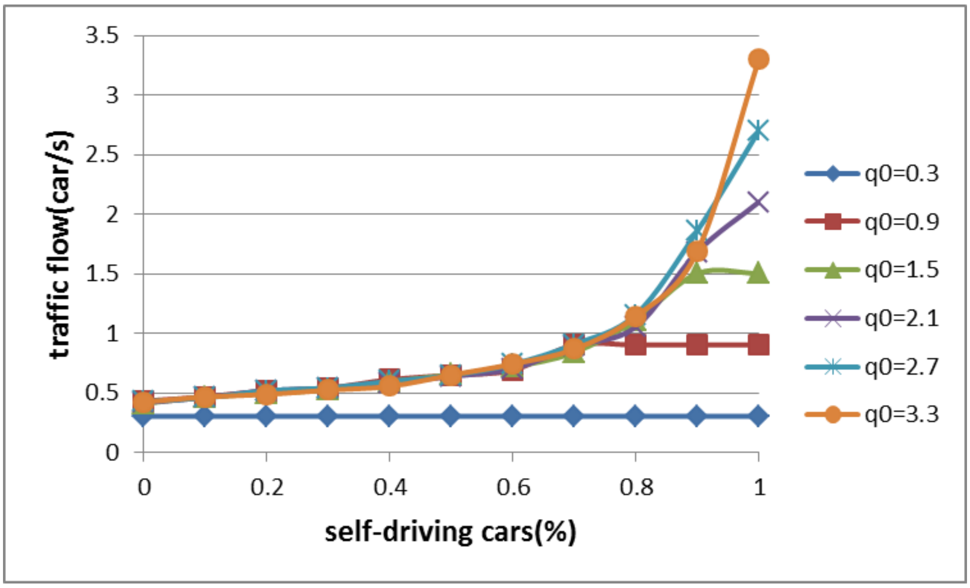


Fig.2 the relationship between mix proportion and balanced flow in the different condition of initial traffic flow (q0 car/s : initial traffic flow)

Fig.2 displays the connection between the percentage of self-driving cars, traffic flow and initial traffic flow. With increase of the percentage of self-driving cars, traffic flow increases. If q0 changes, the connection changes. When $q0 \le 0.3$, traffic flow remains the same, we could see that increase percentage is ineffective for the road which is smooth. When $0.3 \le q0 \le 1.5$, traffic flow increases before they remain the same. We could see that equilibria and tipping point all exist. Equilibria point is the location where traffic flow become maximum in the first time, even if percentage increases, traffic flow remains the same. Tipping point is the location where the slope becomes maximum. When q0=0.9, equilibria and tipping point are 70% and 60%; when q0=1.5, equilibria and tipping point are 90% and 70%. When q0>1.5, traffic flow increases and the two points are inexistent.

We could explain the atmosphere. When q0 is few, traffic flow has been smooth, so traffic flow almost remains the same when the percentage increases; when q0 becomes more, traffic jams comes up, which could be eliminated completely; when q0 increases a lot, the traffic jams won't be eliminated until P=1.

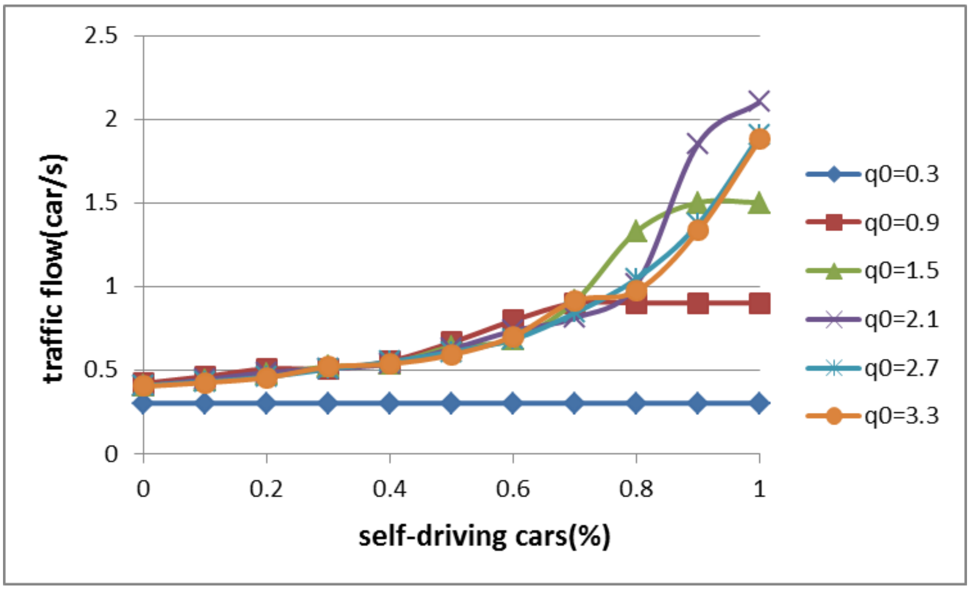


Fig.3 the relationship between mix proportion and balanced flow in the different condition of initial traffic flow (without connection between self-driving cars)

Fig.3 is the same as Fig.1 but self-driving cars not cooperating. From comparison between Fig.2 and Fig.3, we could get some conclusions. When $q0 \le 1.5$, the curve is same from each other; When q0>1.5, the traffic flow of Fig.3 is significantly less than Fig.1. The conclusions tell us that cooperating cars could improve the traffic conditions efficiently.

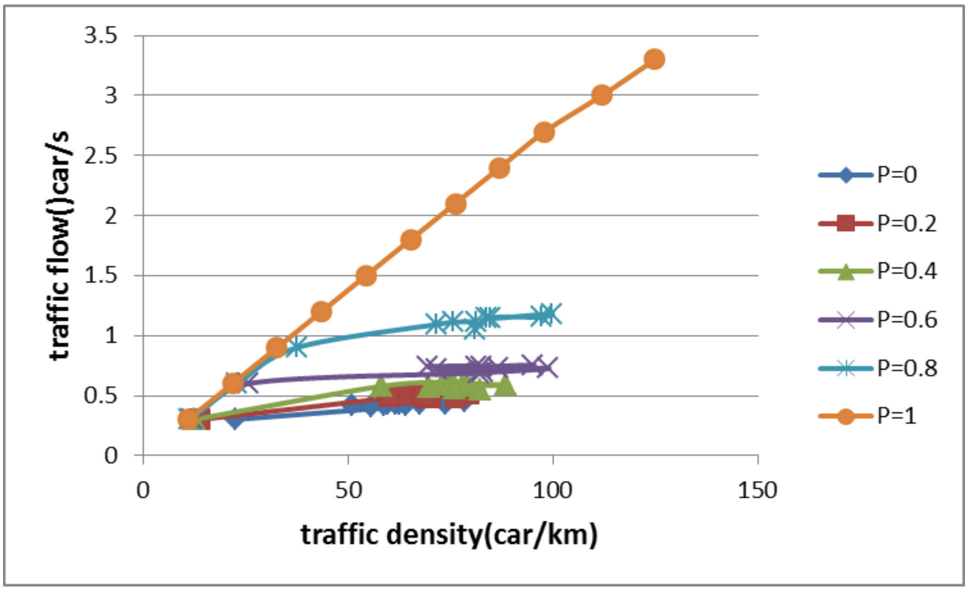


Fig.4 the relationship between traffic density and balanced flow in the different condition of mix proportion

Fig.4 displays that traffic flow increases with the increase of traffic density. According to q=vk, if k increases, q will increase accordingly, but q only increases a little because of decrease of v. But the condition will change greatly when P=1, traffic flow is proportional to the traffic density, the v remains the same even if traffic density changes in a wide range.

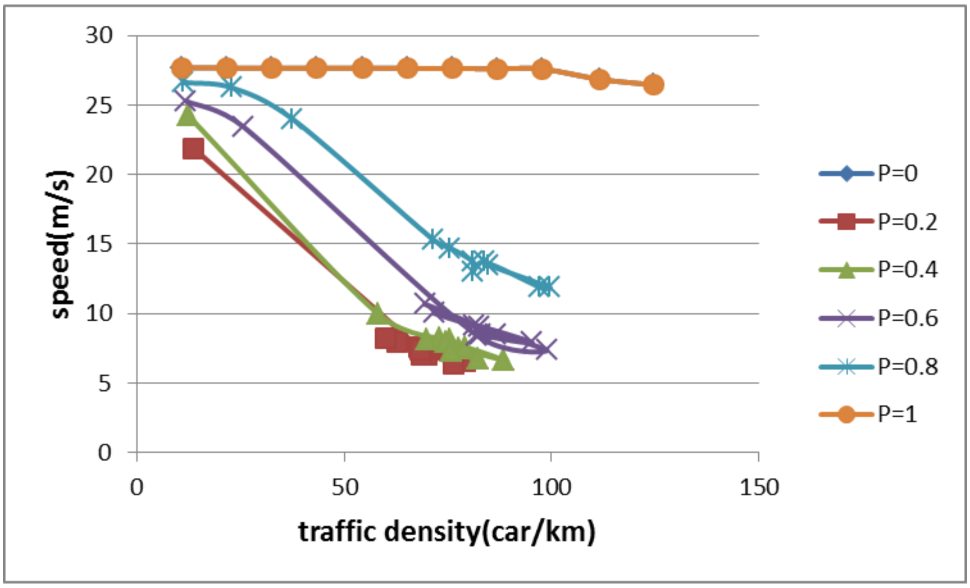


Fig.5 the relationship between traffic density and speed in the different condition of mix proportion (P: percentage of self-driving car)

Fig.5 displays that the speed will decrease with the increase of traffic density. Under the condition of the same vehicle density, speed will increase with the increase of P. When

$P \leq 0.4$, the speed almost remains the same even if P changes in a wide range; when P>0.4, the speed will increase, and the speed will always be fast if P=1.

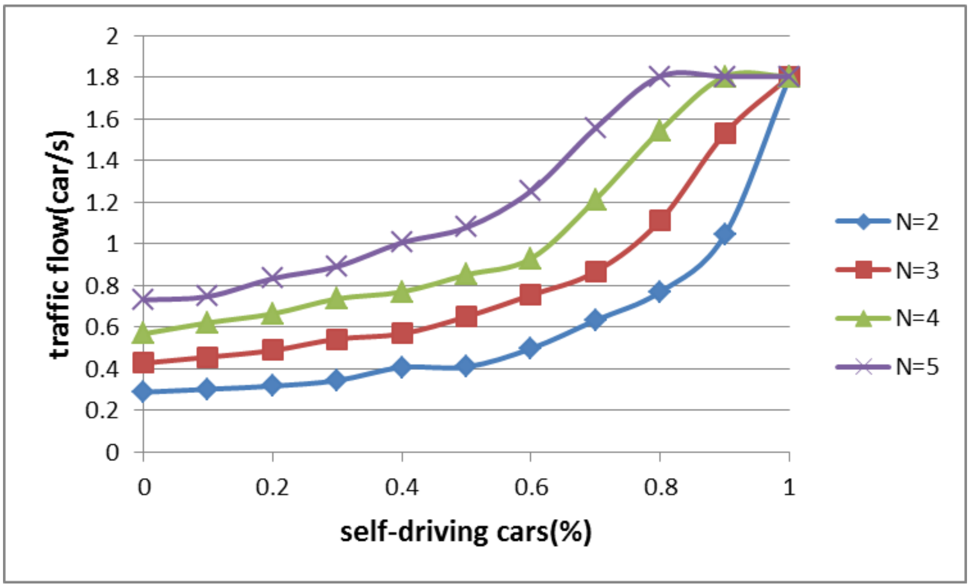


Fig.6 the relationship between mix proportion and traffic flow in the different condition of the number of lanes when q0 = 1.8 (N: number of lanes)

Fig.6 displays the connection between the percentage of self-driving cars, traffic flow and number of lanes. With the increase of percentage, traffic flow increases. When self-driving cars become 100%, their traffic flow all becomes 1.8 car/s. But with the increase of number of lanes, the equilibria point becomes small. Moreover, if the percentage is in the same, traffic flow is proportional to the number of lanes approximately.

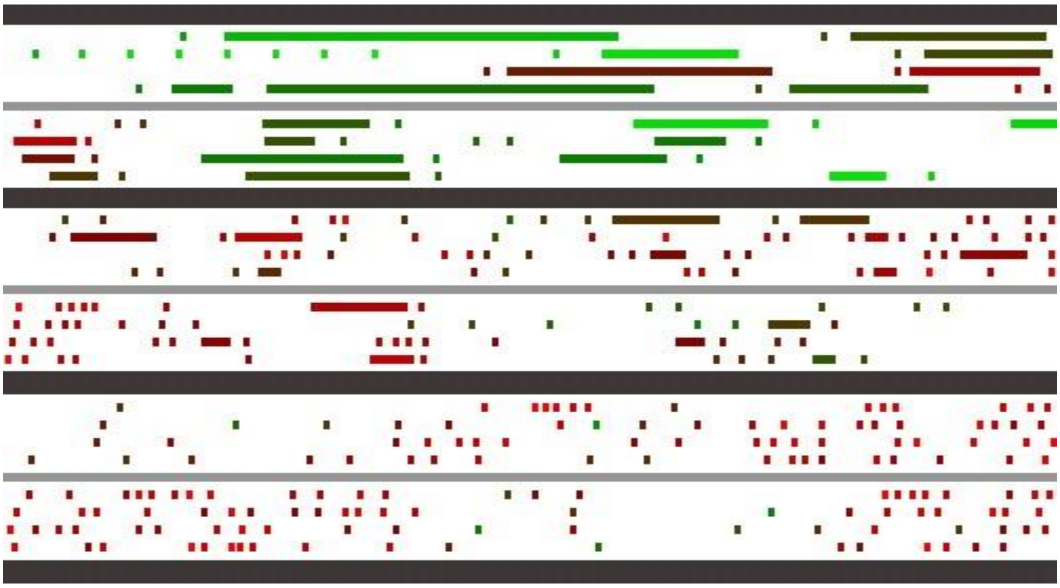


Fig.7 shows different traffic flows when percent of self-driving cars changes

Fig.7 There are three typical situations of traffic flow when the percentage of self-driving cars varies from 10% to 50% to 90%. Three roads in the illustration are the same, respectively including 4 lanes of each direction. Those colored squares represent cars, some of which cooperating with adjacent ones assemble and appear as long rectangles. To show the traffic flow more specifically, we use different colors to reflect speed level, which means color changes from red to green as speed varies from 0 to maximum value. So when the percent is 10%, the illustration below shows that the traffic is still busy and there is little cooperation between self-driving cars. But on the middle road, whose percent is up to 50%, more cars communicate which ease some pressure. And in the upper one, the effect of the self-driving, cooperating system is obvious, high speed cars taking up most of the traffic flow.

4 Data processing and model application

4.1Data Preprocessing

Because the data given in the appendix Excel is too specific and massive to analyze, we should first simplify it. Segment roads according to the intersections provided in the table1, which means, routes between intersections and intersections are classified into one category and the average daily traffic counts in the same category are averaged, as the picture shown in figure 1. [180] means the average traffic count is 180,000. And we can find each new section basically has the same number of lanes.

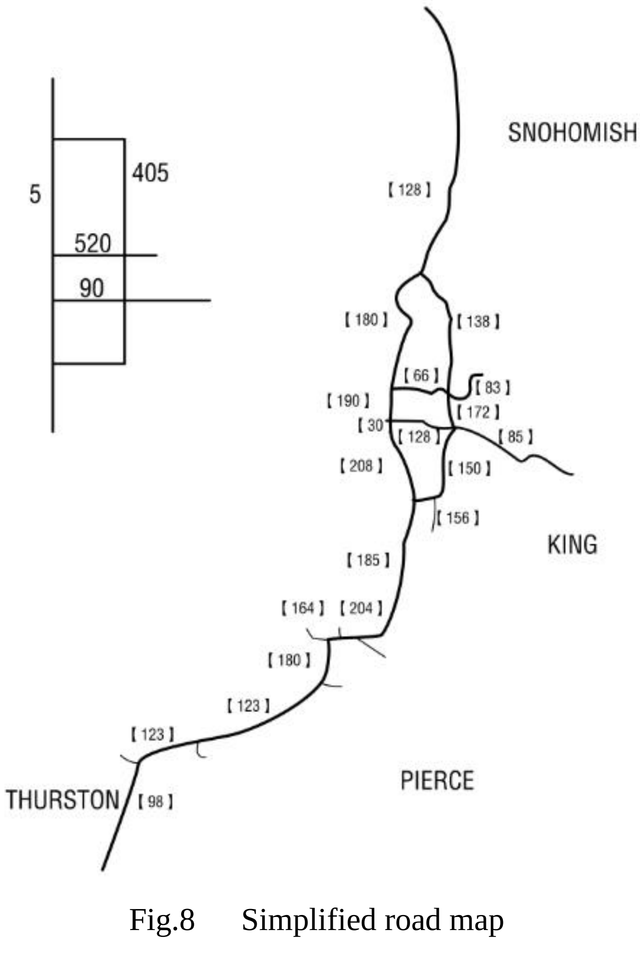


Fig.8 Simplified road map

Data following:

Table 1: Average daily flow in new sections

Route_ID (New)	Average daily traffic counts Year_2015	start comment	end comment
5A	9800	Olympia	Rte 101 intersection
5B	122857	Rte 101 intersection	SR 510 intersection
5C	127789	SR 510 intersection	SR 512 intersection
5D	180500	SR 512 intersection	SR 16 Intersection
5E	164000	SR 16 Intersection	I705 intersection
5F	204333	I705 intersection	SR167 Joins
5G	185666	SR167 Joins	I405 intersection
5H	207636	I405 intersection	I90 intersection
5I	189714	I90 intersection	SR 520 intersection
5J	181826	SR 520 intersection	I405 intersection
5K	127967	I405 intersection	Snohomish County
90A	30000	downtown Seattle	Intersection with I5
90B	128111	Intersection with I5	Intersection with I405
90C	85083	Intersection with I405	/
405A	156000	/	/
405B	150466	/	/
405C	172600	/	/
405D	138705	/	/
520A	66250	/	/
520B	83000	/	/

4.2.Model application

As 8% of the daily traffic volume occurring during peak travel hours, and the hypothesis that those cars concentrate in one hour every day, we can assume an initial flow. Using the initial flow volume in the model as well as changing the percentage of self-driving cars, we can get the actual cost time to transport all cars from peak hours and the traffic congestion rate (counting vehicles whose speed is less than 10Km/h).

4.2.1 Solution of problem 2

In a certain number of lanes, peak/average traffic volume, as well as the case of cooperative interaction in problem 1, when changing the percentage of self-driving cars from 10% to 90%, the time needed for traffic flow and the traffic congestion rate above these sections in different sections of the peak period can be obtained from the model.

Table 2: Peak driving time under different percentage

Section name	Peak driving time under different percentage of self-driving cars.(unit:h)			
	0%	10%	50%	90%
5A	0.79	0.79	0.79	0.78
5B	2.33	2.15	1.58	1.00
5C	3.25	3.08	2.28	1.01
5D	3.52	3.17	2.32	1.05
5E	3.60	3.34	2.42	1.03
5F	4.44	4.14	3.04	1.22
5G	2.84	2.65	1.89	1.00
5H	4.03	3.69	2.67	1.12
5I	3.48	3.25	2.34	1.01
5J	3.62	3.38	2.40	1.07
5K	3.30	3.14	2.23	1.04
90A	1.22	1.20	0.99	0.99
90B	3.26	3.07	2.28	1.03
90C	2.21	1.99	1.51	1.00
405A	4.01	3.71	2.70	1.17
405B	5.88	5.67	4.00	1.70
405C	3.78	3.60	2.48	1.11
405D	3.62	3.44	2.32	1.06
520A	2.72	2.42	1.77	1.00
520B	3.33	2.97	2.25	1.01

Table 3: Road congestion rate under different percentage

Section name	Road congestion rate under different percentage of self-driving cars.			
	0%	10%	50%	90%
5A	0.00%	0.01%	0.00%	0.00%
5B	23.41%	22.30%	14.22%	0.36%
5C	28.60%	28.49%	14.71%	1.00%
5D	24.91%	21.40%	14.92%	1.57%
5E	22.69%	20.53%	11.80%	1.34%
5F	24.56%	22.70%	14.47%	2.41%
5G	25.81%	23.58%	13.89%	0.07%
5H	28.37%	22.63%	13.22%	2.17%
5I	22.83%	21.54%	13.22%	1.15%
5J	22.29%	21.02%	14.65%	2.08%
5K	25.49%	25.74%	15.57%	1.01%
90A	20.70%	18.10%	2.70%	0.00%
90B	29.21%	22.62%	13.42%	2.28%
90C	21.95%	27.01%	15.91%	0.00%
405A	30.91%	26.79%	15.84%	1.48%
405B	28.22%	30.39%	15.77%	2.35%

405C	26.62%	22.92%	13.12%	1.52%
405D	24.81%	20.82%	13.10%	0.88%
520A	29.72%	22.46%	18.58%	0.09%
520B	26.54%	26.02%	22.23%	0.07%

From the above data, we can find when the proportion of self-driving cars increases from 10% to 50%, and then from 50% to 90%, all of the passage time will be greatly shortened, and the congestion rate will also fall sharply, even to 0%. In addition, when the percent increases to 50% , the sections 5A and 90A reach the optimal state with the shortest passsing time, and basically with no congestion.

4.2.2 Solution of problem 3

In the peak hours, there is an equilibria between the time of each section and the proportion of self-driving cars, and select the most representative sections to illustrate the problem, as shown in figure 1:

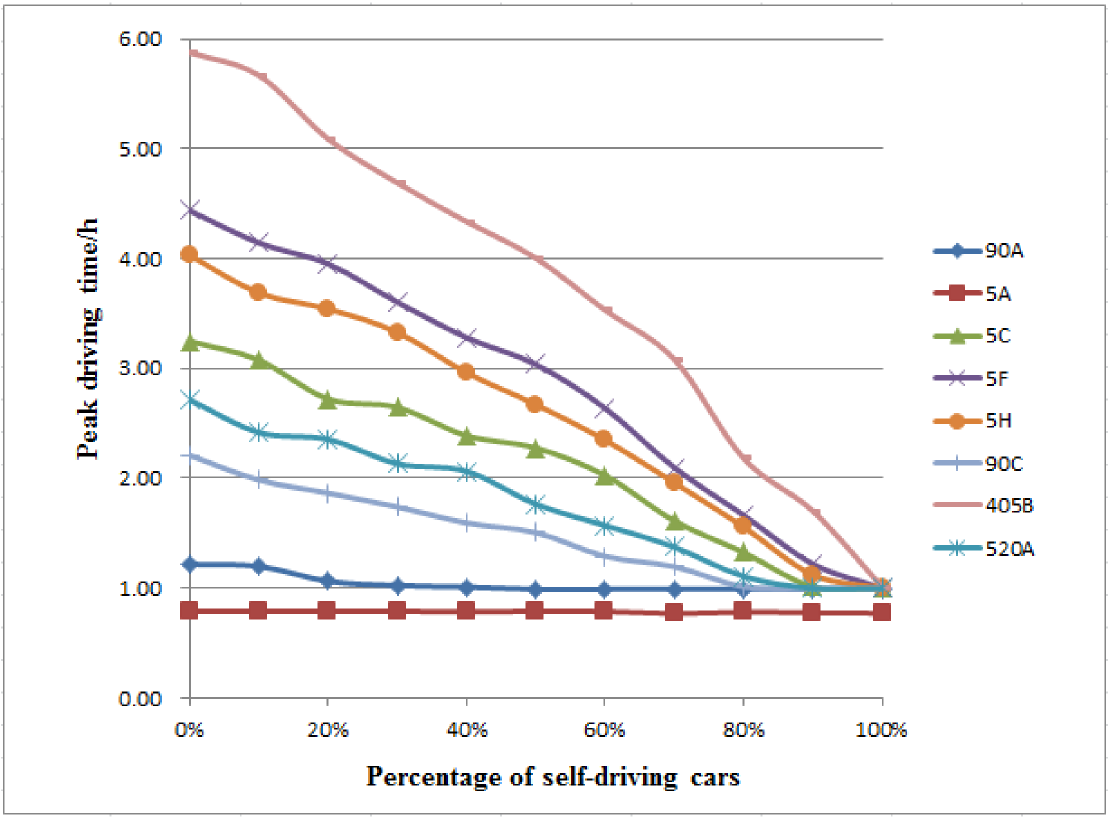


Fig.9 Peak driving time under different percentage of self-driving cars

When the time reduced to half of the initial one, we consider it as the equilibrium point which can be seen in the figure. But the equilibrium point is different from saturation point. The equilibrium point of the percentage if self-driving car respectively are: 90A:15%; 5A:0%; 5C:55%; 5F:58%; 5H:52%; 90C:40%; 405B:58%; 520A:46%. The equilibrium points of the rest sections are: 5B:50%; 5D:52%; 5E:53%; 5G:49%; 5I:52%; 5J:54%; 5K:48%; 90B:55%; 90C:40%; 405A:54%; 405C:53%; 405D:50%; 520B:52%.The distribution of equilibrium points are as follows:

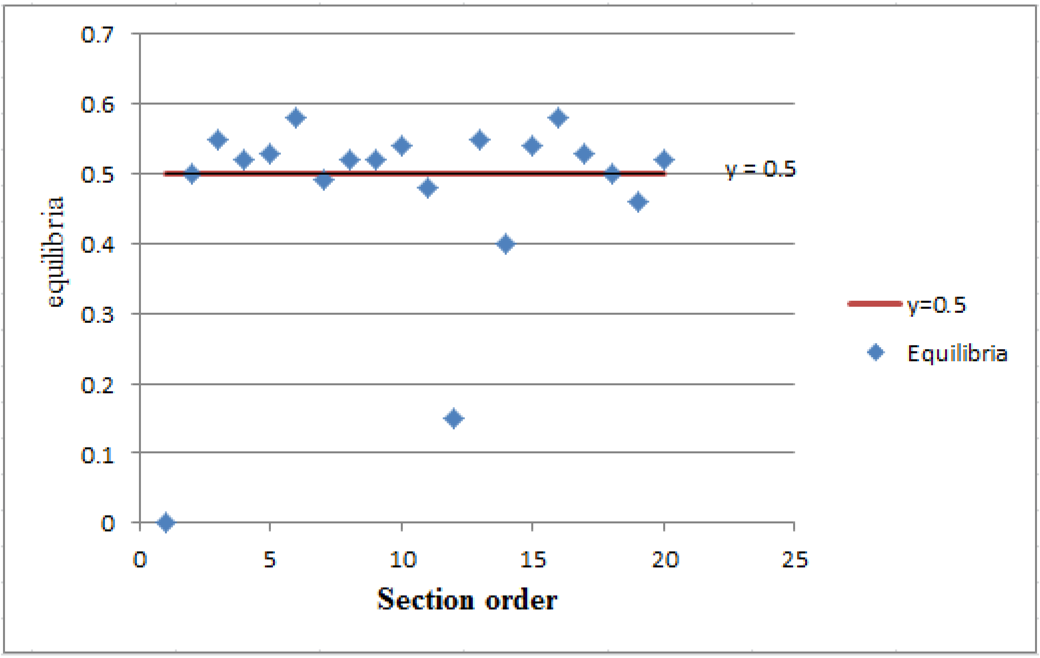


Fig.10 Distribution of equilibrium points in each section

This shows that, apart from a few special sections, the equilibrium points of all other sections vary from 50% to 60%. So for all roads, the percentage of self-driving cars is between 50% and 60%.

4.2.3 Solution of problem 4

There is no mutation point in the curves of peak passage time and the proportion of self-driving cars. But according to the distribution of saturation points in Figure 1, the curves of 90A and 5A in the peak period is relatively smooth, so there is no need to use self-driving cars. While to the curve of 405B, the proportion of 100% also does not help to reach saturation trend, therefore, the lanes of this section should be dedicated to self-driving cars, but other sections do not need to set up dedicated self-driving lanes.

5 Model Checking and Sensitivity Analysis

The model we adopt is the improvement of four traffic models based on cellular automaton. After comparing the test data and the data from literature, and then compared with the actual traffic situation, we can draw a conclusion that our four improved model is indeed feasible, accurate and effective, and it can simplify and explain the problem well.

The following is Sensitivity analysis and stability of the model:

Analysis following: for 90A and 5A, the peak traffic time increases with the proportion of self-driving cars basically changes little, so for what small disturbance the travel time is almost the same, indicating that these two sections have been in a steady state, so the sensitivity is very low.

For the other sections, we select two representative sections to analyze, 5H and 405B. Respectively make 1% disturbance for the percent of self-driving cars, and calculate the data. The different percentage of self-driving cars for 5H and 405B with the corresponding rate of peak traffic time changing compared with the assumed time are following:

Table4 Sensitivity analysis

1% disturbance of the percentage of self-driving cars	Variation of time before and after disturbance	
	Section 5H	Section 405B
10%±1%	7.75%	3.78%
20%±1%	4.01%	8.08%
30%±1%	11.81%	0.14%
40%±1%	5.88%	5.46%
50%±1%	3.74%	7.41%
60%±1%	2.01%	1.69%
70%±1%	1.15%	6.24%
80%±1%	5.05%	8.40%
90%±1%	10.91%	13.24%
100%±1%	0.04%	2.63%

From the table, it can be seen that: for sections like 5H whose saturation point is at 100%, the sensitivity is the least at the equilibrium point (52%), so the model is relatively stable in these areas. But at other percents, it’s more sensitive and instable. For those sections like 405B, who cannot reach saturation point even when percentage is 100%, the sensitivity at the equilibrium point (58%) is the smallest as well as at somewhere like 30%. In other proportions, including 100%, the sensitivity is high, which means the model is not that stable.

6 Conclusions and Discussion

6.1 Conclusions

This paper manages to develop a model to find out the connection among percentage of self-driving cars, traffic flow, traffic density, speed and number of lanes. After that, we apply our model to the data for the roads of interest.

The conclusions below:

1. Because self-driving cars could communicate with each other, the adjacent vehicles have the same motion state. And the distance between adjacent self-driving cars could be ignored. There is safety distance between self-driving cars and the car in the existing traffic flow.
2. The traffic flow usually increases with the increase of percentage of self-driving cars. But there are some special cases. If the initial flow is quite rare, the traffic flow remains the

same. If the initial flow is quite a lot, the traffic flow only becomes maximum when P=1.If the initial flow is between the two before, the equilibria point and tipping point exist. The traffic flow increases before they are the same.

3. The equilibria point could tell us the best percentage of self-driving car for the road. So the lanes should be dedicated to these cars when (equilibria point)=100%.
4. After applying our model to the data for the roads of interest, we work out the peak traffic time of the different percentage of self-driving cars and the different road. The increase of percentage eases traffic congestion effectively. According to the range of equilibria point, government could provide 50%-60% self-driving cars to the road.

6.2Limitation and extensions

Though our model successfully comes up with connection among several traffic variables, it can be improved in several ways:

1. Every car in our model has the same length, and we can introduce different length to the model.
2. Because self-driving car could cooperate with each other, and we can introduce prediction of traffic flow to the model.

References

[1] Qi WANG, Zhiheng LI, Danya YAO, Yi ZHANG, Li LI, & Jianmin HU (2014).Analysis on Road Capacity for Mixed Manual/Automated Traffic.
CICTP 2014: Safe, Smart, and Sustainable Multimodal 351 Transportation Systems © ASCE 2014, 351-358.
[2] Arnab Bose & Petros Ioannou (2001).Mixed Manual/Semi-Automated Traffic: A Macroscopic Analysis.California Partners for Advanced Transportation Technology UC Berkeley.
[3] Gipps P.G. (1981). A behavioural car- following model for computer simulation[J]. Transportation Research Part B: Methodological, 1981, 15(2): 105-111.
[4] Ciuffo B, Punzo V & Montanino M (2012). Thirty years of Gipps' car-following model[J]. Transportation Research Record: Journal of the Transportation Research Board, 2012, 2315(1): 89-99.
[5] Panwai S & Dia H (2005). Comparative evaluation of microscopic car- following behavior[J]. Intelligent Transportation Systems, IEEE Transactions on, 2005, 6(3): 314-325.
[6] Nagel K & Schreckenberg M (1992). A cellular automaton model for free freeway traffic[J]. J Phys I (France) ,1992, 2: 2221-2229.
[7] Rickert M, Nagel K, Schreckenberg M & Latour A (1996). Two lane traffic simulation using cellular automata[J]. Physical A, 1996 , 231: 534-553.
[8] Ebersbach A, Schneider J & Morgenstern I (2001). Simulating traffic on German highways based on the Nagel-Scherckenberg-model[J]. Int .J.M.Phys C , 2001,12(7): 1081-1089.
[9] Yuan Y. M., Jiang R, Hu M. B., Wu Q. S. & Wang R (2009). Physical A, 388-2483.
[10] Davis L. C. (2004). Phys. Rev. E-69-066110.
[11] Jiang R, Wu Q. S. (2005). Eur. Phys. J. B-46-581.
[12] Knorr F & Schreckenberg M (2012) Physical A-391-2225.

Appendix

1.Data preprocessing

Route_ID	Average daily traffic counts Year_2015	Number of Lanes DECR MP direction	Number of Lanes INCR MP direction	average flow(car/day)	needed flow(car/s)	peak car counts	start comment	end comment
5	6,500	3	3	5013	1.3925	0.4	/	/
5	9800	3	3	15680	4.355555556	0.4	Olympia	Rte 101 intersection
5	14400	3	4	4373	1.214722222	0.4	/	/
5	122857	4	4	30933	8.5925	0.4	Rte 101 intersection	SR 510 intersection
5	8900	3	3	5333	1.481388889	0.4	/	/
5	127789	3	3	78400	21.77777778	0.4	SR 510 intersection	SR 512 intersection
5	110000	4	4	4960	1.377777778	0.4	/	/
5	180500	4	4	17547	4.874166667	0.4	SR 512 intersection	SR 16 Intersection
5	134000	4	3	8213	2.281388889	0.4	/	/
5	164000	4	3	4107	1.140833333	0.4	SR 16 Intersection	I705 intersection
5	170000	4	3	2773	0.770277778	0.4	/	/
5	204333	4	3	6080	1.688888889	0.4	I705 intersection	SR167 Joins
5	205000	4	4	6347	1.763055556	0.4	/	/
5	185666	5	5	86453	24.01472222	0.4	SR167 Joins	I405 intersection
5	224000	5	5	13120	3.644444444	0.4	/	/
5	207636	4	4	48213	13.3925	0.4	I405 intersection	I90 intersection
5	149000	2	3	5707	1.585277778	0.4	/	/
5	189714	4	4	13387	3.718611111	0.4	I90 intersection	SR 520 intersection
5	157000	4	4	2720	0.755555556	0.4	/	/
5	181826	4	4	69333	19.25916667	0.4	SR 520 intersection	I405 intersection

5	158000	3	3	6827	1.896388889	0.4	/	/
5	127967	3	3	175573	48.77027778	0.4	I405 intersection	northern boundary of Snohomish County
5	62000	3	3	15040	4.177777778	0.4	/	/
90	13000	2	3	533	0.148055556	7.2	/	/
90	30000	2	2	4000	1.111111111	7.2	Start of I90 in downtown Seattle	Intersection with I5
90	121000	4	4	6133	1.703611111	7.2	/	/
90	128111	3	3	41067	11.4075	7.2	Intersection with I5	Intersection with I405
90	101000	3	3	3733	1.036944444	7.2	/	/
90	85083	3	3	69493	19.30361111	7.2	Intersection with I405	/
405	75000	3	3	480	0.133333333	32.4	/	/
405	156000	3	3	11573	3.214722222	32.4	/	/
405	93000	3	3	2880	0.8	32.4	/	/
405	150466	2	2	43360	12.04444444	32.4	/	/
405	102000	4	4	4053	1.125833333	32.4	/	/
405	172600	4	3	16747	4.651944444	32.4	/	/
405	130000	3	3	4960	1.377777778	32.4	/	/
405	138705	3	3	77067	21.4075	32.4	/	/
405	35000	3	3	587	0.163055556	32.4	/	/
520	48000	2	2	1920	0.533333333	41.6	/	/
520	66250	2	2	35040	9.733333333	41.6	/	/
520	109000	2	2	14240	3.955555556	41.6	/	/
520	83000	2	2	640	0.177777778	41.6	/	/

2. The blocking rate on each way

Route name	Initial flow	Blocking rate under percentage				
		0	0.1	0.2	0.3	0.4
5A	0.22	6.00E-06	0.000118582	0	7.00E-06	0.00118541
5B	2.73	0.234094	0.222977	0.188948	0.171678	0.149526
5C	2.84	0.285986	0.28494	0.207688	0.205956	0.192706
5D	4.01	0.249075	0.213955	0.205437	0.18497	0.20814

5E	3.64	0.226894	0.205287	0.189555	0.195332	0.157914
5F	4.54	0.245641	0.227004	0.231165	0.196174	0.167406
5G	4.12	0.258093	0.235841	0.213097	0.145697	0.147399
5H	4.61	0.28365	0.226289	0.234586	0.165328	0.186695
5I	4.21	0.22834	0.21541	0.180377	0.222728	0.137335
5J	4.04	0.222878	0.21017	0.212846	0.204277	0.16437
5K	2.84	0.254903	0.257352	0.214965	0.219832	0.196954
90A	0.67	0.207	0.181	0.113	0.074	0.101
90B	2.85	0.292074	0.226218	0.222943	0.211167	0.15341
90C	1.89	0.219549	0.270134	0.224277	0.20746	0.136289
405A	3.47	0.309114	0.26785	0.236713	0.237405	0.16472
405B	3.34	0.282209	0.303887	0.220819	0.218982	0.220306
405C	3.84	0.266152	0.229232	0.188781	0.18575	0.168465
405D	3.08	0.248085	0.208243	0.203314	0.175512	0.201388
520A	1.47	0.297151	0.224599	0.228136	0.209083	0.20016
520B	1.84	0.265359	0.260199	0.196574	0.232554	0.182134

0.5	0.6	0.7	0.8	0.9	1
0	0	0	0	0	0
0.142189	0.113154	0.0972247	0.0168242	0.00357153	0
0.147086	0.115706	0.0645856	0.0470318	0.0099683	0
0.149219	0.0953406	0.0824152	0.0541296	0.0156611	0
0.117972	0.114707	0.0820065	0.0398744	0.0133654	0
0.144708	0.100265	0.0908087	0.0568717	0.0240735	0
0.138924	0.116181	0.0848964	0.05783	0.000676633	0
0.132238	0.121243	0.0881533	0.0631714	0.0217433	0
0.132201	0.123987	0.0767546	0.0746115	0.0115203	0
0.146526	0.130822	0.0912029	0.0451527	0.0208478	0
0.155687	0.132847	0.074131	0.0426594	0.0100803	0
0.027	0.006	0	0	0	0
0.134199	0.132576	0.110306	0.0586889	0.0227636	0
0.159059	0.0909227	0.068463	0.0115643	0	0
0.158391	0.146185	0.104075	0.0670038	0.014786	0
0.157683	0.158492	0.121562	0.0575618	0.0234931	0
0.131155	0.156239	0.0967709	0.0725137	0.0152343	0
0.131032	0.154983	0.0859613	0.0765272	0.00882821	0
0.185803	0.131976	0.0980553	0.0365977	0.000912852	0
0.222332	0.0840703	0.0657324	0.0588987	0.000670316	0

3. The peak time on each way

Route name	Peak flow	Peak time under percentage(unit:*0.1s)				
		0	0.1	0.2	0.3	0.4
5A	784	28509	28509	28509	28509	28303
5B	9828	84000	77385	73233	70099	61195
5C	10223	116967	110758	98015	95274	85979
5D	14440	126889	113970	111937	102484	92091
5E	13120	129516	120256	112425	102260	97619
5F	16347	159794	149151	142271	129738	118114
5G	14854	102159	95524	89643	83684	74643
5H	16610	145065	132773	127475	119841	106611
5I	15177	125396	116929	109615	101013	93124
5J	14546	130498	121805	114284	106729	99066
5K	10238	118632	113127	101870	94884	87955
90A	2400	43875	43165	38338	36866	36418
90B	10249	117534	110441	107657	95606	88735
90C	6806	79602	71491	67186	62670	57386
405A	12480	144277	133618	125175	113764	107586
405B	12037	211546	204016	183211	168821	156121
405C	13808	136173	129652	118523	105889	108298
405D	11096	130234	123701	110407	101798	92853
520A	5300	97785	87027	84800	76811	74333
520B	6640	119855	106924	99104	98079	89487
0.5	0.6	0.7	0.8	0.9	1	
28509	28405	27703	28201	28000	27801	
56776	49964	43448	36548	35934	35921	
81980	72969	58052	47726	36484	35945	
83419	70199	61212	51682	37741	36054	
87176	75057	64630	50813	37156	35994	
109344	94985	75505	59923	43967	35982	
67919	60902	50335	43205	35992	36000	
96178	84529	70560	56058	40393	35991	
84080	74138	60406	48583	36501	35960	
86429	77711	60902	52789	38569	36007	
80361	68711	56501	47508	37324	35998	
35714	35714	35714	35714	35714	35714	
82123	72023	58902	46187	37242	35911	
54231	46489	43075	36493	35934	35934	
97120	85420	69837	55888	42219	35924	
144155	127106	110940	78519	61288	36409	
89141	81415	62906	54170	40139	35911	

83428	78527	64064	52888	38039	35979
63549	56563	49486	39879	36005	36005
80876	65807	58041	51234	36284	36047

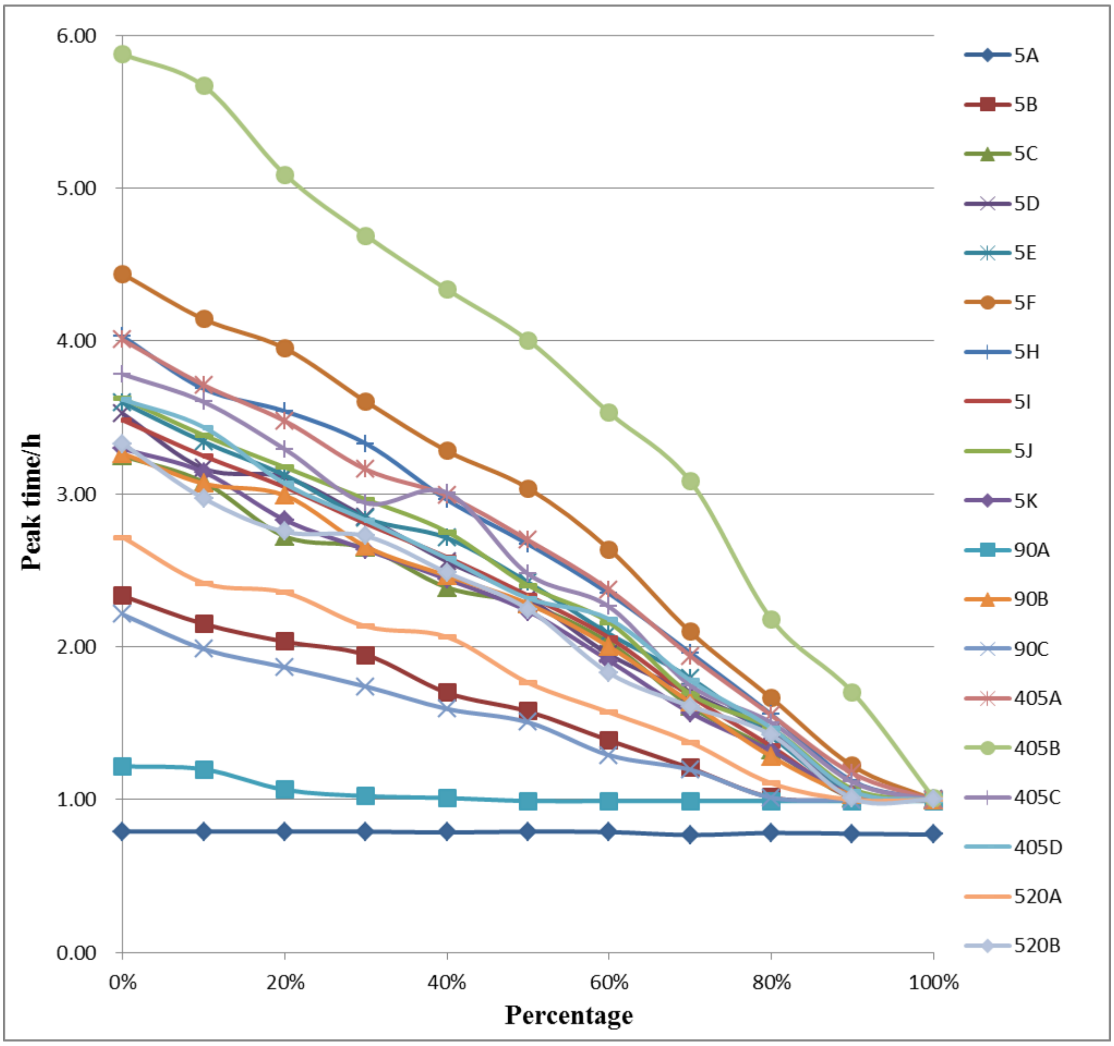
4. The peak time on each way

Route name	Peak time under percentage(unit:h) 0	0.1	0.2	0.3	0.4	0.5
5A	0.79	0.79	0.79	0.79	0.79	0.79
5B	2.33	2.15	2.03	1.95	1.70	1.58
5C	3.25	3.08	2.72	2.65	2.39	2.28
5D	3.52	3.17	3.11	2.85	2.56	2.32
5E	3.60	3.34	3.12	2.84	2.71	2.42
5F	4.44	4.14	3.95	3.60	3.28	3.04
5G	2.84	2.65	2.49	2.32	2.07	1.89
5H	4.03	3.69	3.54	3.33	2.96	2.67
5I	3.48	3.25	3.04	2.81	2.59	2.34
5J	3.62	3.38	3.17	2.96	2.75	2.40
5K	3.30	3.14	2.83	2.64	2.44	2.23
90A	1.22	1.20	1.06	1.02	1.01	0.99
90B	3.26	3.07	2.99	2.66	2.46	2.28
90C	2.21	1.99	1.87	1.74	1.59	1.51
405A	4.01	3.71	3.48	3.16	2.99	2.70
405B	5.88	5.67	5.09	4.69	4.34	4.00
405C	3.78	3.60	3.29	2.94	3.01	2.48
405D	3.62	3.44	3.07	2.83	2.58	2.32
520A	2.72	2.42	2.36	2.13	2.06	1.77
520B	3.33	2.97	2.75	2.72	2.49	2.25

0.6	0.7	0.8	0.9	1
0.79	0.77	0.78	0.78	0.77
1.39	1.21	1.02	1.00	1.00
2.03	1.61	1.33	1.01	1.00
1.95	1.70	1.44	1.05	1.00
2.08	1.80	1.41	1.03	1.00
2.64	2.10	1.66	1.22	1.00
1.69	1.40	1.20	1.00	1.00
2.35	1.96	1.56	1.12	1.00
2.06	1.68	1.35	1.01	1.00
2.16	1.69	1.47	1.07	1.00
1.91	1.57	1.32	1.04	1.00
0.99	0.99	0.99	0.99	0.99

2.00	1.64	1.28	1.03	1.00
1.29	1.20	1.01	1.00	1.00
2.37	1.94	1.55	1.17	1.00
3.53	3.08	2.18	1.70	1.01
2.26	1.75	1.50	1.11	1.00
2.18	1.78	1.47	1.06	1.00
1.57	1.37	1.11	1.00	1.00
1.83	1.61	1.42	1.01	1.00

5.The relationship with peak time and percentage on each way



6. The equilibria on each way

Route name	equilibria	Route name	equilibria
5A	0	5K	0.48
5B	0.5	90A	0.15
5C	0.55	90B	0.55
5D	0.52	90C	0.4
5E	0.53	405A	0.54
5F	0.58	405B	0.58
5G	0.49	405C	0.53
5H	0.52	405D	0.5
5I	0.52	520A	0.46
5J	0.54	520B	0.52

7. The interval of probability changes into 1%

(1) Section 5H:

percent	perCongestion	time	percent	perCongestion	time
0	0.188065	143313	0.51	0.14716	96122
0.01	0.190575	141723	0.52	0.166174	95022
0.02	0.188304	139932	0.53	0.128246	94697
0.03	0.244722	140287	0.54	0.126769	89397
0.04	0.250835	137046	0.55	0.12129	86196
0.05	0.232388	139228	0.56	0.115326	88116
0.06	0.226922	136147	0.57	0.106703	85266
0.07	0.194922	134930	0.58	0.087764	84357
0.08	0.184143	136933	0.59	0.116627	83635
0.09	0.186835	138186	0.6	0.137597	84357
0.1	0.216513	130274	0.61	0.0902732	83383
0.11	0.215292	132456	0.62	0.129651	84058
0.12	0.208071	130581	0.63	0.101872	79702
0.13	0.209879	132140	0.64	0.0992533	80670
0.14	0.185424	130684	0.65	0.0873058	79284
0.15	0.204227	128064	0.66	0.0825396	75124
0.16	0.191493	125738	0.67	0.121076	75226
0.17	0.210025	125833	0.68	0.114713	79019
0.18	0.178497	121684	0.69	0.0951973	74019

0.19	0.185288	125453	0.7	0.108691	73625
0.2	0.172573	122492	0.71	0.0759439	73171
0.21	0.174747	120537	0.72	0.0684134	64882
0.22	0.211412	125547	0.73	0.0824365	66573
0.23	0.184946	117968	0.74	0.0669944	65060
0.24	0.164968	116398	0.75	0.0705455	66096
0.25	0.188742	118812	0.76	0.0702274	62303
0.26	0.201271	121152	0.77	0.0721404	63518
0.27	0.189703	117885	0.78	0.0582105	60007
0.28	0.175151	115507	0.79	0.0483651	56864
0.29	0.18205	115749	0.8	0.0672013	55719
0.3	0.183011	122042	0.81	0.04229	54051
0.31	0.169176	113923	0.82	0.0467898	53615
0.32	0.160765	112002	0.83	0.0303778	51599
0.33	0.179464	113147	0.839999	0.0479793	52052
0.34	0.201264	109564	0.849999	0.0327671	50609
0.35	0.168473	107161	0.859999	0.0480964	49243
0.36	0.133637	105527	0.869999	0.0414086	48910
0.37	0.194014	107647	0.879999	0.0360428	45871
0.38	0.158294	105594	0.889999	0.0301794	46280
0.39	0.181605	106542	0.899999	0.0299723	42997
0.4	0.149506	103167	0.909999	0.0246279	41587
0.41	0.162875	105863	0.919999	0.0243628	40981
0.42	0.148527	101901	0.929999	0.00722018	37827
0.43	0.163027	100788	0.939999	0.00799869	36505
0.44	0.162971	103424	0.949999	0.00126954	36211
0.45	0.173779	101157	0.959999	0.000969543	36093
0.46	0.155189	103231	0.969999	0.000947506	36100
0.47	0.138065	97077	0.979999	0.000122574	36077
0.48	0.123955	98575	0.989999	4.92E-07	36077
0.49	0.124603	97591	0.999999	0	36069
0.5	0.139321	95077	1.01	0	36077

(2) Section 405B

percent	perCongestion	time	percent	perCongestion	time
0	0.281773	207892	0.51	0.16881	141279
0.01	0.319366	210437	0.52	0.101863	134642
0.02	0.290905	210437	0.53	0.138123	133596
0.03	0.229441	206821	0.54	0.179019	130979

0.04	0.270481	199618	0.55	0.176296	127106
0.05	0.257029	203327	0.56	0.117592	135399
0.06	0.256887	206466	0.57	0.136722	130694
0.07	0.240125	201963	0.58	0.140771	122701
0.08	0.230064	198958	0.59	0.170434	120490
0.09	0.219238	201963	0.6	0.106606	120974
0.1	0.210106	204710	0.61	0.131294	119414
0.11	0.235124	209703	0.62	0.122736	120853
0.12	0.22849	192592	0.63	0.112925	115740
0.13	0.186831	198303	0.64	0.115855	117091
0.14	0.238044	189261	0.65	0.0846108	118009
0.15	0.257761	196683	0.66	0.123238	121955
0.16	0.23147	188963	0.67	0.0836436	111764
0.17	0.23286	189858	0.68	0.109211	118591
0.18	0.270763	189559	0.69	0.092714	112600
0.19	0.249946	183211	0.7	0.132318	106427
0.2	0.221578	186620	0.71	0.0971931	106900
0.21	0.242256	174956	0.72	0.0795291	98744
0.22	0.230987	180194	0.73	0.116394	100224
0.23	0.20955	190157	0.74	0.0845964	103056
0.24	0.268953	179122	0.75	0.0841736	95455
0.25	0.240749	177536	0.76	0.098782	92592
0.26	0.194647	181007	0.77	0.126804	93673
0.27	0.223604	170254	0.78	0.0642443	94704
0.28	0.193486	176237	0.79	0.0681426	87605
0.29	0.186244	172697	0.8	0.0441645	83590
0.3	0.229657	172945	0.81	0.0525719	86597
0.31	0.220295	172945	0.82	0.0446622	80948
0.32	0.16152	168821	0.83	0.0892713	80839
0.33	0.17186	170737	0.839999	0.0310674	73485
0.34	0.201029	169296	0.849999	0.0324532	73575
0.35	0.186337	160066	0.859999	0.0394915	69860
0.36	0.197052	154320	0.869999	0.0321389	69257
0.37	0.235661	161787	0.879999	0.0419248	72424
0.38	0.16358	153925	0.889999	0.033579	68044
0.39	0.231799	156527	0.899999	0.0327946	63856
0.4	0.192484	152367	0.909999	0.0190644	59589
0.41	0.173077	156527	0.919999	0.0186013	55778
0.42	0.153957	153533	0.929999	0.015849	52085
0.43	0.164624	148421	0.939999	0.0140279	48263

0.44	0.180535	157346	0.949999	0.00811954	46189
0.45	0.164936	150839	0.959999	0.00460528	42443
0.46	0.169325	150839	0.969999	0.00209219	40257
0.47	0.108746	143468	0.979999	0.00149371	39017
0.48	0.237753	147693	0.989999	0.000134161	37197
0.49	0.154903	142113	0.999999	0	36299
0.5	0.183823	147151	1.01	0	36354

8. The code (win64 \ visual studio2015 \ opencv3.1)

```
#include<fstream>
#include<iostream>
#include<list>
#include<opencv2\highgui\highgui.hpp>
#include<opencv2\imgproc\imgproc.hpp>
#include<opencv2\core\core.hpp>
#include <opencv2\opencv.hpp>
#include<cstdlib>
#include<ctime>
#include<string>
#include<streambuf>

using namespace cv;
using namespace std;

int main(int argc, char **argv)
{

	int higLength =4000;
	float flow = 6;
	float percent = 0.0;
	float proSlow = 0.4;
	float proChange = 0.5;
	int allTime = 10000;
	const int a = 10;
	const int b = 20;
	const int vmax = 90;
	const float delT = 0.1;
	struct Car
	{
		int react = 10;
		int mark = 10;
		int froPosition = 12;
		int froDistance = 1000;
		int safDistance = 0;
```

```
        int speed = vmax;
        int safSpeed = 0;
        int numbers = 1;
        int length = 12;
    };
    ofstream oFile;
    oFile.open("time.csv", ios::out | ios::trunc);
    //oFile << "percent" << "," << "flow" << "," << "perCongestion" << "," << "carNum" <<
endl;
    oFile << "percent" << "," << "perCongestion" << "," << "time" << endl;
    int count;
    cout << "time:";
    cin >> allTime;
    cout << "flow:";
    cin >> flow;
    cout << "car number:";
    cin >> count;
    const int lefLane =4;
    const int rigLane =4;
    for (flow; flow < 3.3; flow += 0.3)
    {
        cout << "---------------flow" << flow << "-------------------" << endl;
        for (percent = 0.0; percent < 1.1; percent += 0.1)
        {
            list<Car>lefCar[lefLane];
            list<Car>rigCar[rigLane];
            srand((unsigned)time(0));
            int timer = 0;
            int number = 0;
            float congestion = 0;
            float sum = 0;
            while (true)
            {
                timer++;

                if ((int)((timer - 1)*delT*flow) != (int)(timer*delT*flow))//creat new car
according to traffic flow
                {
                    Car newCar;

                    if (rand() % 100 / (float)100 < percent)
                    {
                        newCar.react = 1;//self-driving
                        newCar.mark = 1;
```

```
            }
            int lane = rand() % (lefLane + rigLane);
            //int lane = rand() % (lefLane);
            if (lane < lefLane)
            {
                int maxLane = 0;
                bool bChange = false;
                for (int laneNum = 0; laneNum < lefLane; laneNum++)
                {
                    if (lefCar[laneNum].empty())
                    {
                        lefCar[laneNum].push_back(newCar);//put into left lanes
                        number++;
                        break;
                    }
                    else if (lefCar[laneNum].back().froPosition -
lefCar[laneNum].back().length > vmax/3)
                    {
                        if (lefCar[laneNum].back().froPosition -
lefCar[laneNum].back().length >
                            lefCar[maxLane].back().froPosition -
lefCar[maxLane].back().length)
                            maxLane = laneNum;
                        bChange = true;
                    }
                }
                if (bChange)
                {
                    lefCar[maxLane].push_back(newCar);//put into left lanes
                    number++;
                }
            }
            else
            {
                int maxLane = 0;
                bool bChange = false;
                for (int laneNum = 0; laneNum < rigLane; laneNum++)
                {
                    if (rigCar[laneNum].empty())
                    {
                        rigCar[laneNum].push_back(newCar);//put into left lanes
                        number++;
                        break;
                    }
```

```
                    else if (rigCar[laneNum].back().froPosition -
rigCar[laneNum].back().length > vmax/3)
                    {
                        if (rigCar[laneNum].back().froPosition -
rigCar[laneNum].back().length >
                            rigCar[maxLane].back().froPosition -
rigCar[maxLane].back().length)
                            maxLane = laneNum;
                        bChange = true;
                    }
                }
                if (bChange)
                {
                    rigCar[maxLane].push_back(newCar);//put into left lanes
                    number++;
                }
            }
        }
        for (int i = 0; i < lefLane + rigLane; i++)
        {
            bool isLeft = true;
            list<Car>::iterator index;
            list<Car>nowLane;
            if (i < lefLane)
            {
                nowLane = lefCar[i];
                index = lefCar[i].begin();
            }
            else
            {
                nowLane = rigCar[i - lefLane];
                index = rigCar[i - lefLane].begin();
                isLeft = false;
            }
            int frontPosition = 2 * higLength;
            int frontSpeed = vmax;
            int frontLength = 12;
            //for (list<Car>::iterator count = nowLane.begin(); count !=
nowLane.end(); ++index, ++count)
            for (index; isLeft&&index != lefCar[i].end() ||
                (!isLeft) && index != rigCar[i - lefLane].end(); ++index)
            {
                if ((*index).mark == (*index).react)
                {
```

```
(*index).froDistance = frontPosition - (*index).froPosition -
frontLength;
(*index).safDistance = (*index).speed*delT*(*index).react +
    (*index).speed*(*index).speed / (2 * b) -
    frontSpeed*frontSpeed / (2 * b);

(*index).safSpeed = -b*delT*(*index).react +
    sqrtf(b*b*delT*delT*(*index).react*(*index).react +
        frontSpeed*frontSpeed + 2 * b*(*index).froPosition);
if ((*index).froDistance > (*index).safDistance)
{
    (*index).mark = 0;
    (*index).speed = max(0,
        min(min(int((*index).speed + a*delT*(*index).react),
vmax),
            min((*index).safSpeed, (*index).froDistance)));
}
else if ((*index).froDistance < (*index).safDistance)
{

    bool isDcre = false;
    if((*index).react == 1 &&
        (((isLeft && (index) != lefCar[i].begin()) ||
        (!isLeft) && (index) != rigCar[i - lefLane].begin())
        && (isDcre = true) && (*--index).react == 1))
    {

        list<Car>::iterator temp = index;
        index++;
        if ((*temp).length + (*index).length <
(*temp).froPosition)
        {
            (*temp).length = ((*temp).froPosition -
                (*index).froPosition + (*index).length);
            (*temp).numbers += (*index).numbers;
            if (isLeft)
            {
                lefCar[i].erase(index);
            }
            else
            {
                rigCar[i - lefLane].erase(index);
            }
            index = temp;
```

```
                        (*index).mark = 0;
                    }
                }
                else if (rand() % 100 / (float)100 < proChange)
                {
                    if (isDcre)index++;
                    if (isLeft)
                    {
                        list<Car>::iterator indexLeft;
                        list<Car>::iterator indexRight;
                        bool isChangeLeft = false;
                        bool isChangeRight = false;
                        if (i - 1 >= 0)
                        {
                            for (indexLeft = lefCar[i - 1].begin();
                                indexLeft != lefCar[i - 1].end();
indexLeft++)
                            {
                                if ((*indexLeft).froPosition -
(*indexLeft).length -
                                    (*index).froPosition >
(*index).speed)
                                    if (((++indexLeft) == lefCar[i -
1].end() ||
                                        (*index).froPosition -
(*indexLeft).froPosition > 2 * (*index).length))
                                    {
                                        isChangeLeft = true;
                                        indexLeft--;
                                        break;
                                    }
                                    else
                                        indexLeft--;
                            }
                        }
                        if (i + 1<lefLane)
                        {
                            for (indexRight = lefCar[i + 1].begin();
                                indexRight != lefCar[i + 1].end();
indexRight++)
                            {
                                if ((*indexRight).froPosition
-(*indexRight).length-
                                    (*index).froPosition >
```

```
(*index).speed)
                                        if  (((++indexRight)  ==  lefCar[i  +
1].end() ||
                                            (*index).froPosition                -
(*indexRight).froPosition > 2 * (*index).length))
                                        {
                                            isChangeRight = true;
                                            indexRight--;
                                            break;
                                        }
                                        else
                                            indexRight--;
                                    }
                                }
                                if                (isChangeRight)//                &&
(*indexRight).froPosition>(*indexLeft).froPosition)
                                {
                                    list<Car>::iterator temp = index;
                                    temp--;
                                    (*index).mark = 0;
                                    lefCar[i + 1].insert(++indexRight, *index);
                                    lefCar[i].erase(index);
                                    index = temp;
                                }
                                else if (isChangeLeft)
                                {
                                    list<Car>::iterator temp = index;
                                    temp--;
                                    (*index).mark = 0;
                                    lefCar[i - 1].insert(++indexLeft, *index);
                                    lefCar[i].erase(index);
                                    index = temp;
                                }
                                else
                                {
                                    (*index).mark = 0;
                                    (*index).speed            =            max(0,
min((*index).froDistance, (*index).safSpeed));
                                }
                            }
                            else
                            {
                                list<Car>::iterator indexLeft;
                                list<Car>::iterator indexRight;
```

```
bool isChangeLeft = false;
bool isChangeRight = false;
if (i - lefLane - 1 >= 0)
{
    for (indexLeft = rigCar[i - lefLane -
1].begin();
        indexLeft != rigCar[i - lefLane - 1].end();
indexLeft++)
    {
        if ((*indexLeft).froPosition -
(*indexLeft).length-
            (*index).froPosition >
(*index).speed)
            if (((++indexLeft) == rigCar[i -
lefLane - 1].end() ||
                (*index).froPosition -
(*indexLeft).froPosition > 2 * (*index).length))
            {
                indexLeft--;
                isChangeLeft = true;
                break;

            }
            else
                indexLeft--;
    }
}
if (i - lefLane + 1<rigLane)
{
    for (indexRight = rigCar[i - lefLane +
1].begin();
        indexRight != rigCar[i - lefLane +
1].end(); indexRight++)
    {
        if ((*indexRight).froPosition -

    (*indexRight).length-(*index).froPosition > (*index).speed)
            if (((++indexRight) == rigCar[i -
lefLane + 1].end() ||
                (*index).froPosition -
(*indexRight).froPosition > 2 * (*index).length))
            {
                isChangeRight = true;
                indexRight--;
```

```
                                        break;

                                    }
                                    else
                                        indexRight--;
                                }
                            }
                            if
(isChangeRight)//&&(*indexRight).froPosition>(*indexLeft).froPosition)
                            {
                                list<Car>::iterator temp = index;
                                temp--;
                                (*index).mark = 0;
                                rigCar[i - lefLane + 1].insert(++indexRight,
*index);

                                rigCar[i - lefLane].erase(index);
                                index = temp;
                            }
                            else if (isChangeLeft)
                            {
                                list<Car>::iterator temp = index;
                                temp--;
                                (*index).mark = 0;
                                rigCar[i - lefLane - 1].insert(++indexLeft,
*index);

                                rigCar[i - lefLane].erase(index);
                                index = temp;
                            }
                            else
                            {
                                (*index).mark = 0;
                                (*index).speed = max(0,
min((*index).froDistance, (*index).safSpeed));
                            }
                        }
                    }
                    else
                    {
                        if (isDcre)index++;
                        (*index).mark = 0;
                        (*index).speed = max(0, min((*index).froDistance,
(*index).safSpeed));
                    }
                }
```

```
                }
                else
                {
                    (*index).mark++;
                }
                if ((*index).react == 10 && rand() % 100 / (float)100 < proSlow)
                    (*index).speed = max((int)((*index).speed - b*delT), 0);
                (*index).froPosition = (*index).froPosition + (*index).speed*delT;

                frontSpeed = (*index).speed;
                frontPosition = (*index).froPosition;
                frontLength = (*index).length;
            }
        }

        Mat highway(higLength, 2*lefLane + 2*rigLane + 3, CV_8UC3, Scalar(255,
255, 255));
        for (int r = 0; r < highway.rows; r++)
        {
            highway.at<Vec3b>(r, highway.cols/2)[2] = 150;
            highway.at<Vec3b>(r, highway.cols / 2)[1] = 150;
            highway.at<Vec3b>(r, highway.cols / 2)[0] = 150;
        }
        for (int i = 0; i < lefLane; i++)
        {
            while (!lefCar[i].empty() && lefCar[i].front().froPosition -
lefCar[i].front().length > higLength)
            {
                lefCar[i].pop_front();
            }
            for (list<Car>::iterator index = lefCar[i].begin(); index !=
lefCar[i].end(); ++index)
            {
                Car car = *index;
                int frontRow = car.froPosition;
                for (int k = frontRow - car.length; k < min(frontRow, higLength);
k++)
                {
                    highway.at<Vec3b>(k, 2*i + 1)[2] = max(0, 255 - car.speed *
4);

                    highway.at<Vec3b>(k, 2*i + 1)[1] = max(0, (car.speed - 30) *
4);

                    highway.at<Vec3b>(k, 2 * i + 1)[0] = 0;// car.speed ;
                }
```

```
					sum += car.numbers;
					if (car.speed < 10)
						congestion += car.numbers;
				}
			}

			for (int i = 0; i < rigLane; i++)
			{
				while (!rigCar[i].empty() && rigCar[i].front().froPosition - 
rigCar[i].front().length > higLength)
				{
					rigCar[i].pop_front();
				}
				for (list<Car>::iterator index = rigCar[i].begin(); index != 
rigCar[i].end(); ++index)
				{
					Car car = *index;
					int frontRow = car.froPosition;
					for (int k = max(higLength - frontRow, 0); k < higLength - 
frontRow + car.length; k++)
					{
						highway.at<Vec3b>(k, 2 * lefLane + 2 * rigLane + 1 - 2 * i)[2] 
= max(0, 255 - car.speed * 4);
						highway.at<Vec3b>(k, 2 * lefLane + 2 * rigLane + 1 - 2 * i)[1] 
= max(0, (car.speed - 30) * 4);
						highway.at<Vec3b>(k, 2 * lefLane + 2 * rigLane + 1 - 2 * i)[0] 
= 0;// car.speed ;
					}
					sum += car.numbers;
					if (car.speed < 10)
						congestion += car.numbers;
				}
			}

			resize(highway, highway, Size(highway.cols * 6, highway.rows / 4),0,0,0);
			cv::imshow("highway", highway);
			cv::waitKey(1);

			if (number > count)
			{
				sum /= timer;
				congestion /= timer;
				float perCongestion = congestion / sum;
				cout << "percent:" << percent << "," << perCongestion << "," << timer
```

```
<< endl;
					oFile << percent << "," << perCongestion << "," << timer << endl;
					break;
				}
				if (timer == allTime)
				{
					sum /= allTime;
					congestion /= allTime;
					float perCongestion = congestion / sum;
					timer = timer*count / number;
					cout << "percent:" << percent << "," << perCongestion << "," << timer
<< endl;
					oFile << percent << "," << perCongestion << "," << timer << endl;
					break;
				}
			}
		}
	}
	system("pause");
}
```