For office use only	Team Control Number	For office use only
T1 _____	**58900**	F1 _____
T2 _____		F2 _____
T3 _____	Problem Chosen	F3 _____
T4 _____	**B**	F4 _____

2017
MCM/ICM
Summary Sheet

Turnpike Toll Plaza Model Based on Queuing Theory

Summary

Based on queuing theory, we build several models to simulate the traffic condition on the toll plaza on New Jersey Turnpike. First, we break the process into two parts, entering the toll station and entering the toll plaza. Accordingly, we build up two models, inflow model and merging queue model. These two models can help us generate the incoming cars and calculate the throughput when vehicles exit the toll plazas.

Then we design the models in detail. Inflow model considers the number of vehicles which enter the tolls. Merging queue model calculates the maximum number of vehicles in toll plaza, average time consumed when vehicles are merging lanes, the time needed to drive through the toll plaza, and finally the throughput at the exit. We use basic physic kinetic laws and mathematic laws to calculate them.

We also design two new toll plazas according to the calculation, separated toll plaza and reversible toll plaza. These two designs improve the efficiency of the toll plazas without increasing the area.

Meanwhile, we simulate the traffic condition with the help of MATLAB and C++. By developing computer programs, we get the solution in different conditions, including light or heavy traffic, accidents, self-driving vehicles and electronic tollbooths. We change the parameters and get the throughput in different conditions. Then, we compare the results to get the conclusions.

In sum, we develop and evaluate two toll plazas: reversible plaza and separate plaza. Compared with traditional toll plaza, reversible plaza performs much better in extreme conditions like accidents, while separate plaza occupies smaller area. ItâĂŹs worth well improving the traditional toll plaza into the two new designs.

Keywords: Toll Plaza; Queuing Theory; Inflow Model; Merging Queue Model; Throughput; Poisson Distribution

Turnpike Toll Plaza Model Based on Queuing Theory

Team #58900

January 24, 2017

Summary

Based on queuing theory, we build several models to simulate the traffic condition on the toll plaza on New Jersey Turnpike. First, we break the process into two parts, entering the toll station and entering the toll plaza. Accordingly, we build up two models, inflow model and merging queue model. These two models can help us generate the incoming cars and calculate the throughput when vehicles exit the toll plazas.

Then we design the models in detail. Inflow model considers the number of vehicles which enter the tolls. Merging queue model calculates the maximum number of vehicles in toll plaza, average time consumed when vehicles are merging lanes, the time needed to drive through the toll plaza, and finally the throughput at the exit. We use basic physic kinetic laws and mathematic laws to calculate them.

We also design two new toll plazas according to the calculation, separated toll plaza and reversible toll plaza. These two designs improve the efficiency of the toll plazas without increasing the area.

Meanwhile, we simulate the traffic condition with the help of MATLAB and C++. By developing computer programs, we get the solution in different conditions, including light or heavy traffic, accidents, self-driving vehicles and electronic tollbooths. We change the parameters and get the throughput in different conditions. Then, we compare the results to get the conclusions.

In sum, we develop and evaluate two toll plazas: reversible plaza and separate plaza. Compared with traditional toll plaza, reversible plaza performs much better in extreme conditions like accidents, while separate plaza occupies smaller area. ItâĂŹs worth well improving the traditional toll plaza into the two new designs.

Keywords: Toll Plaza; Queuing Theory; Inflow Model; Merging Queue Model; Throughput; Poisson Distribution

1 A Letter to New Jersey Turnpike Authority

Dear New Jersey Turnpike Authority,
It's our great honor to submit our solution to optimize the design of the toll plaza of the barrier tolls. Barrier toll is one of the most important facilities on the toll highway. However, the traditional design of the fan-in area and merging pattern may cause traffic congestion, which may reduce the speed and efficiency of the highway. Moreover, the waiting vehicle will waste more energy and cause air pollution. Therefore, it's necessary to work out a solution that considers all the cost in building toll plaza and the throughput efficiency of both traditional toll plaza and the new design. We will briefly discuss the strategy and model we use and offer you a new design in this letter. Hope that our solution will help you optimize the tolls of the highway in New Jersey.
In our model, we consider these following factors.

1. The shape, size and merging pattern of the fan-in area and the relation with L lanes of travel and B tollbooths.
2. The cost to build the toll considering the land and road construction.
3. Accident prevention and possibility that an accident will cause traffic congestion.
4. The performance of our model in light and heavy traffic.
5. The affect of self-driving vehicles and automatic charging tolls on our model.

The modeling is built based on those factors. We simulate the vehicles entering the toll using queuing theory. Queuing theory is the mathematic model of waiting lines or queues. It can help us simulate the length of the queue and waiting time at the toll. The vehicles arriving at the toll satisfy Poisson distribution. After charging, the vehicles will enter the toll plaza, which is a trapezoid. In the fan-in area, the vehicles will merge together at the end of the toll plaza, which will cause the car to slow down. In heavy traffic condition, the plaza may be crowded with vehicles and the speed of vehicles will be much slower. The numbers of lanes we have will also affect the efficiency to exit the toll. We use the throughput to represent the efficiency of the plaza.
Considering both the cost and efficiency, we work out that the traditional toll plaza is a good design that it's in good efficiency. To promote it, we can separate the big toll into several small tolls, which will save more land and improve the throughput. The small tolls will be arranged separately in the direction of lanes. Reversible tollbooths can also improve the efficiency. When one direction of the toll is in heavy traffic, we can use several tollbooths in opposite direction to increase the number of tollbooths and area of toll plaza.
Hope out solution can benefit the traffic condition in New Jersey!

Best regards,
Team #58900

Contents

2 Introduction

New Jersey has 38131 miles of roads, including 356 miles of toll roads maintained by state agencies. New Jersey turnpike is one of the toll road in the state. The turnpike accesses to Delaware, Pennsylvania and New York. According to the International Bridge, Tunnel and Turnpike Association, the New Jersey turnpike is the sixth busiest toll road and the most heavily traveled highways in the United States. On the turnpike, one of the most important facility is toll. Every entrances and exits have tolls. Barrier tolls also appear on the turnpikes.

The speed limit of this road is $65\ mph$. However, when the vehicles go through the tolls, they will slow down and wait to pay the bill. The separating and merging of the road near tolls will also make the vehicles reduces the speed. On this extremely busy turnpike, tolls seem to be the bottleneck and cause traffic congestion. How can we design the toll plaza to reduce the congestion and improve the efficiency of throughput?

Based on the Poisson distribution, queuing theory and basic kinematics laws, we establish several models to simulate the traffic condition on the toll plaza. Using the model, we work out the best way to simulate the toll plaza and design a better one.

Our model can be divided into 2 sub-models, including inflow and merging queue model. The inflow model employs Poisson distribution and queuing theory, which we can generate the vehicles entering the toll. Using the vehicles generated and the merging queue model, we can calculate the condition for the vehicles to leave and therefore the throughput. Outflow model includes the basic calculation of the relation between the plaza area and L and B. Kinetic simulation of the vehicles to merge is also in the merging queue model. Using these two models, we can analysis the performance of traditional toll plaza as well as the new design. All the other parameters such as accident or self-driving cars can also be considered in these models by changing the assumption of the model.

We use `MATLAB` and C++ to help us acquire and analysis the massive discrete data. We also use the program to help us verify the model.

2.1 Tasks

Our team build up the models mainly based on these following information and requirements.

1. The shape, size and merging pattern of the fan-in area and the relation with L lanes of travel and B tollbooths.
2. The cost to build the toll considering the land and road construction.
3. Accident prevention and possibility that an accident will cause traffic congestion.
4. The performance of our model in light and heavy traffic.
5. The affect of self-driving vehicles and automatic charging tolls on our model.

The goal of this problem is to consider all these factors and analysis the performance of the toll plazas. Moreover, we have to give better solutions if exist.

2.2 Terminology

- **Barrier tolls:** A row of tollbooths placed across the highway, perpendicular to the direction of traffic flow. In this problem, we only consider this kind of toll.
- **Toll plaza:** The place where vehicle leaving tollbooths and before entering the merging area.
- **Merging area:** The area between toll plaza and highway lanes, where vehicles merge from B tollbooths to L lanes.
- **Traditional tolls:** The common design of the toll plazas, which is widely used in most countries. The view from top is as follows

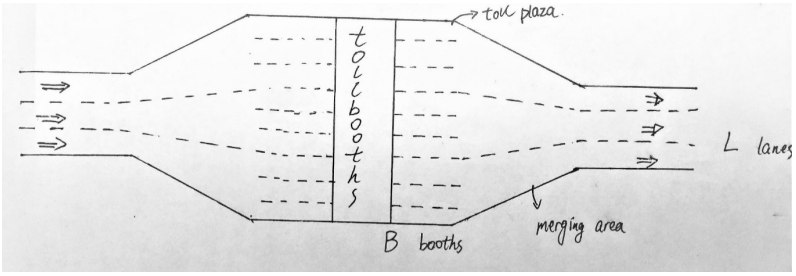

Figure 1: Traditional tolls

- **Separated tolls:** A new design of the toll plazas, which divide a large plaza into 3 or more small plazas, and place them along the direction of the lanes. The view from top is as follows

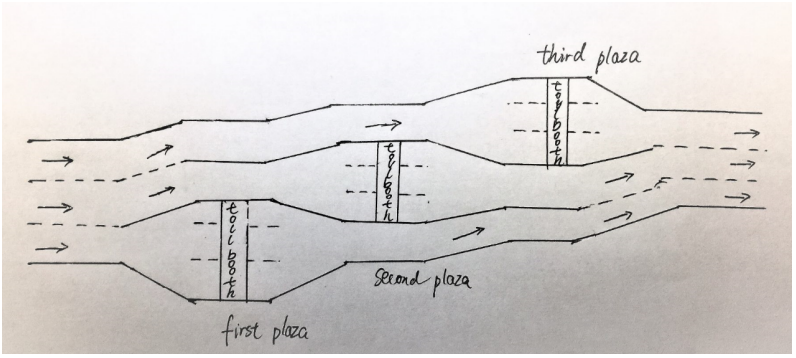

Figure 2: Separated tolls

- **Reversible tolls:** A new design of the toll plazas, when one direction of the vehicles fulfilled the plaza, some tollbooths and lanes in the opposite direction can be used in the congestion direction. The top view is the same as the traditional one.
- **ETC:** Means both exact-change (automated) tollbooths and electronic toll collection booths. Both these two booths can reduce the time drivers use in the toll stations.

2.3 Assumptions

- All the vehicles are in same size. Omit the difference of freight car and sedan car.
- Drivers are rational. They obey the traffic rules.
- The width of one lane is only for one vehicle.
- The environment and time in a day won't affect the driving.
- Vehicles in the toll plaza are in slow speed. Only when the vehicles leave the merging area, drivers will accelerate to $60mph$.

2.4 Notations

Symbol	Explain	Unit
B	numbers of tollbooths	/
L	numbers of lanes	/
N	numbers of vehicles in toll plaza at moment	/
N_{max}	maximum numbers of vehicles in toll plaza at moment	/
D	width of lanes	m
a	length of vehicles	m
b	width of vehicles	m
d	safety distance between vehicles	m
d_0	minimum safety distance between vehicles	m
l	length of toll plaza	m
l_0	length of merging area	m
s	one side width of merging area	m
v_0	vehicle speed in light traffic	m/s
v_1	vehicle speed in heavy traffic	m/s
v_2	vehicle speed when merging	m/s
t_0	time needed in light traffic	s
t_1	time needed in heavy traffic	s
T	serving time at tollbooths	s
Δt	average reaction time of human	s
n	total numbers of merging in the merging area	/
a_1	acceleration	m/s^2
a_2	deceleration	m/s^2
t_d	average time consumed when vehicles are merging	s
ρ	service intensity at tollbooths	/
W	throughput	$1/s$

Table 1: Notations.

3 Models

3.1 Inflow Model

We suppose that vehicles arrive at the tolls randomly. According to Queuing Theory [3], the number of arriving vehicles (or customers in Queuing Theory) obeys Poisson distribution (suppose parameter is λ), the service time obeys exponential distribution (suppose parameter is μ). When a car arrives at tolls, it will immediately find a vacant toll. If no tolls are available, the car will automatically find a toll and wait for service. Accordingly, we define S_n to describe all possible situations. Suppose S_n is the nth situation, P_n is the probability of situation S_n.

If $n > B$, S_n means all the tolls are in use and $n - B$ vehicles are waiting.
If $n \leq B$, S_n means n tolls are in use and no vehicles are waiting.

Suppose $\rho = \lambda/(B \cdot \mu)$. Here $\rho \leq 1$, otherwise vehicles tend to accumulate in the front of tolls. According to Queuing Theory,

$$P_0 = \left[\sum_{k=0}^{B-1} \frac{1}{k!}(\frac{\lambda}{\mu})^k + \frac{1}{B!(1-\rho)}(\frac{\lambda}{\mu})^B\right]^{-1} \tag{1}$$

$$P_n = \begin{cases} \frac{1}{n!}(\frac{\lambda}{\mu})^n \cdot P_0, \ n \leq B \\ \frac{B^B}{B!} \cdot \rho^n \cdot P_0, \ n > B \end{cases}$$

In any moment, suppose the probability of having n tolls in use is P'_n.

If $n < B$,

$$P'_n = P_n \tag{2}$$

If $n \geq B$,

$$P'_n = \sum_{i=B}^{\infty} P_i \tag{3}$$

Therefore, we can sum up equation 1,2,3 and get

$$P_0 = \left[\sum_{k=0}^{B-1} \frac{1}{k!}(\frac{\lambda}{\mu})^k + \frac{1}{B!(1-\rho)}(\frac{\lambda}{\mu})^B\right]^{-1} \tag{4}$$

$$P'_n = \begin{cases} \frac{1}{n!}(\frac{\lambda}{\mu})^n \cdot P_0, \ n < B \\ \frac{B^B}{B!}\frac{\rho^B}{1-\rho}P_0, \ n = B \end{cases}$$

3.2 Merging Queue Model

3.2.1 Basic Assumption

1. All drivers need follow the traffic rules. All the drivers will leave enough space with the car before. When one car wants to merge into another queue, the car behind will slow down and keep a certain distance. They will not speed up maliciously or hit into the moving car.

2. The toll plaza are can be seen as a rectangular and the merge area can be seen as a trapezoid, where the width of the rectangular and the length of upline and base trapezoid are depend on tollbooths B and lanes L. The length of the toll plaze is l_0 and the height of the trapezoid is l.

3. All the cars merge into another queue only if they are driving in the merge point area. They will not change lanes casually before they enter into the merge area. The amount of vehicle merging into another lane is a small part of the total at a certain moment.

4. When the car slow down in order to leave enough space for another car merging, the deceleration will satisfy that the car which reducing speed will still have a speed.

5. All cars in this model are vehicle with same length and width, same speed during normal situation, same acceleration and same deceleration. Here we assume that the speed maintain at a lower level with respect to the speed on the highway.

6. The merging behavior can be seen as a movement in a line with constant speed.

7. Vehicle from different tollbooths will spend different journey to get out of the merge area. In this model, we use the average journey instead of the different value to consider the traffic flow on a macro level.

8. We also consider the average time to reflect the situation of the circulation. Each time vehicle merge into another lane, the delay time is same and it can be calculated based on the deceleration and the velocity of the traffic flow. According to the total number of merging times in the merge area, the average time can be determined and the average speed and be calculated.

9. The response time of different drivers Δt are equal.

10. The safety distance between vehicles depends on the average speed.

11. The average speed of the traffic flow during two adjacent unit time period are similar. Hence, average vehicle speed v_1 at different time can be replaced by the speed during the previous unit time.

3.2.2 Maximum Number of Vehicles in Toll Plaza, N_{max}

In this model, each car will occupy equal area in the toll plaza and merge area. We assume that the number is given by the total area divided by the area that one car may occupy. And we ignore the difference between the reality and the model, since only

some area may lead more vehicle or less vehicle.

1. **Calculation of actual distance between vehicles** d

   The actual distance need satisfy that: when a driver notice that the car before slow down, he will slow down after Δt. The minimum distance during the process should equal to the safety distance.

   Therefore,

$$d = d_0 + v_1 \cdot \Delta t \tag{5}$$

   Notice that, the speed v_1 is deduced from the actual maximum number of vehicle. when we use the computer to simulate the process, we first use the speed during the previous unit time period to calculate the maximum number of vehicle during this time period. Since we assume that average vehicle speed v_1 at different time can be replaced by the speed during the previous unit time. At beginning of the simulation, we input the v_0 as the average speed in the light traffic condition.

2. **Calculation of Maximum number of Vehicle in Toll Plaza**

   Based on our assumption, one car will occupy $(a + d) \cdot D$.
   Then, the maximum number of vehicle in toll plaza can be expressed as

$$N = \frac{B \cdot D \cdot l_0 + (B + L) \cdot D \cdot l \cdot \frac{1}{2}}{D \cdot (a + d)} \tag{6}$$

3.2.3 Average Time Consumed When Vehicles are Merging

Consider the condition, the vehicle which merge into another lane drive into another lane at the speed v_2. Take the car which intend to merge be **A**. Take the vehicle waiting be **B**. **A** prepares to merge when the head of **B** is at the middle of **A**. Then **B** decelerate at a_2 until the distance between **B** and the vehicle in front of **B** is $2d+a$. Then, here we assume that the trajectory of **A** is very similar to a line segment, which length is $\sqrt{D^2 + L^2}$. After **A** merge into another line

1. **Calculation of distance journey between light traffic and heavy traffic**

   The **B** will slow down decelerate at a_2, then it will move at a lower speed until **A** finishes its merging. And then it will accelerate at a_1 until it get the speed v_2.

   The journey **B** moving during the process is

$$v_2 \left(\sqrt{\frac{2(a+d)}{a_2}} + \frac{\sqrt{D^2 + L^2}}{v_2} + \frac{\sqrt{2a_2(a+d)}}{a_1} \right) - (a+d) - \sqrt{2a_2(a+d)} \cdot \frac{\sqrt{D^2 + L^2}}{v_2} \tag{7}$$

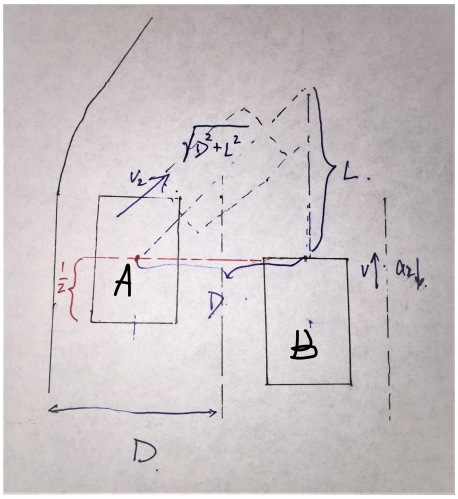

Figure 3: Merging Models

2. **Calculation of The time that one vehicle merge will spend redundant time t_d.**

The whole process will spend

$$\sqrt{\frac{2(a+d)}{a_2}} + \frac{\sqrt{D^2+L^2}}{v_2} + \frac{\sqrt{2a_2(a+d)}}{a_1} \tag{8}$$

The time that one vehicle merge will spend redundant time t_d.

$$t_d = \frac{(a+d)(a_1+a_2)}{a_1 \cdot v_2} + \frac{\sqrt{2a_2 \cdot (a+d) \cdot (D^2+L^2)}}{v_2} \tag{9}$$

3.2.4 Time Needed in Heavy Traffic t_1

1. **Calculation of time needed in light traffic t_0**

First, we calculate the average journey that vehicle from different tollbooths to get the highway enterancce, which can be expressed as

$$\sum_{i=1}^{\lfloor \frac{B}{2} \rfloor} \sqrt{\frac{(B-(2i+1))^2 \cdot D^2}{4} + l^2} + B \cdot l_0 \tag{10}$$

The figure to help us calculating the average journey is shown below.

In this condition, all vehicles will travel at speed v_0, then the time needed in light traffic is

$$t_0 = \sum_{i=1}^{\lfloor \frac{B}{2} \rfloor} \frac{\sqrt{\frac{(B-(2i+1))^2 \cdot D^2}{4} + l^2} + B \cdot l_0}{B \cdot v_0} \tag{11}$$

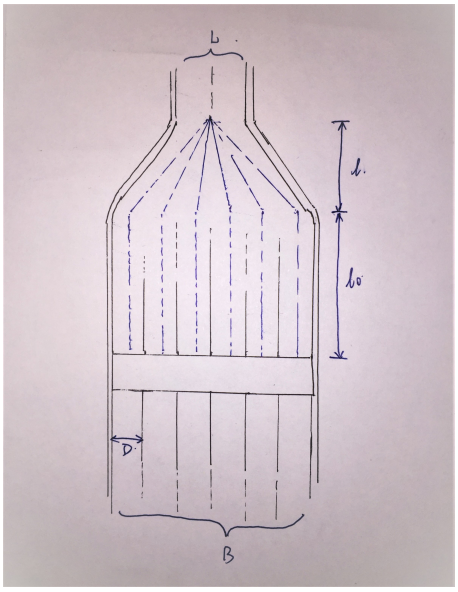

Figure 4: Calculation of average journey

2. **total numbers of merging in the merging area**

   Total numbers of merging in the merging area depends on B,L and N.

$$n = \left\lceil \frac{N \cdot (B - L)}{L} \right\rceil \tag{12}$$

3. **Calculation of time needed in heavy traffic** t_1

   Consider all vehicles in the toll plaza and merge area, the average time can be given by

$$t_1 = \frac{n \cdot t_d + N \cdot t_0}{N} \tag{13}$$

3.2.5 The Calculation of Throughput

Based on the time needed in heavy traffic t_1, we can get the average vehicle speed in heavy traffic v_1.

$$v_1 = \sum_{i=1}^{\lfloor \frac{B}{2} \rfloor} \frac{\sqrt{\frac{(B-(2i+1))^2 \cdot D^2}{4} + l^2} + B \cdot l_0}{B \cdot t_1} \tag{14}$$

Therefore, the total throughput is

$$W = \frac{v_1}{a + d} \cdot L \tag{15}$$

4 Design of Two New Tolls

According to the models we made and the expectation of the results, we find that in the normal conditions, the toll plaza will hold very few vehicles, and the throughput is large enough for all the vehicles to exit the plaza. It means that the traditional plaza is too large. However, considering the width of the tollbooths and safety reasons, we can't just reduce the area of toll plazas and merging area, for we have to follow the instruction of toll plaza construction and leave enough space for vehicles to merge. Therefore, we have to work out a new kind of toll plaza to reduce the consuming land. Or we can improve the efficiency of the toll plaza by not enlarging the area. Then we thought of these following two design.

4.1 Separated Tolls

To reduce the consuming of land, we divide a large toll into several small ones. Place all the small tolls along the direction of the lanes. The design is shown in the graph below.

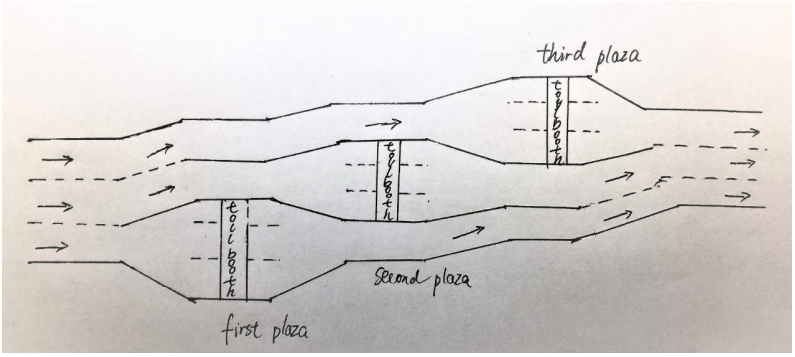

Figure 5: Separated tolls

From the figure, we can calculate and find that each area of small plaza can be reduced.

4.2 Reversible Tolls

When one direction of the vehicles fulfilled the plaza, some tollbooths and lanes in the opposite direction can be used in the congestion direction. Although we cannot increase the numbers of booths, lanes or plaza area, we can increase the access ability of one side, then the congestion will be reduced.

5 Model Analysis

Based on **Inflow Model** and **Merging Queue Model**, we use `MATLAB` to simulate the input traffic flow [1] and operate the input data through *C++* to calculate the throughput

[1] The `MATLAB` code is given in the **Appendix**

W [s^{-1}]. Then, we use the average of throughput in 2000[s] to represent the specific throughput at certain period. The input data also has a quality, service intensity ρ, which determined by the expectation of *Poisson distribution* controlled by a parameter in MATLAB. We use service intensity ρ to determine light or heavy traffic. ρ mustn't be higher than 1 or vehicles will accumulate in front of tolls. So we used $\rho = 1$ as heavy traffic. In our following simulation, we find that both our plaza and traditional one can handle heavy traffic, so it is easier for them to handle light traffic. Other constants and analog emerency accident are shown in the table below.

D	3.2 [m]	a	4 [m]	d_0	1 [m]	Δt	0.5 [s]
v_0	10 [m/s]	v_2	10 [m/s]	a_1	2 [$m \cdot s^{-2}$]	a_2	2 [$m \cdot s^{-2}$]

	B - L	l_0 [m] [2]	l [m]	Accident Treatment
Traditional Tolls	9-3	100	60	Emergency is handled after 11 hours
Separated Tolls (single)	3-1	30	20	Borrow a single separate toll from the oppsite direction after 30 minutes and emergency is handled after 11 hours

Table 2: Constant Used in this Model and Accident Treament (1.5 exit roads are strucked after 5 hours)

Besides, those two tolls a have a huge difference by floor area. In order to estimate the floor area, we still follow the assumption of the merging quene model. That is, the floor area A can be expressed as

$$A = B \cdot D \cdot l_0 + (B + L) \cdot D \cdot l \cdot \frac{1}{2}$$

We substitude the relative amout into the expression, the we can get

$$A_{traditional} = 9 \times 3.2 \times 100 + (3 + 9) \times 3.2 \times 60 \times \frac{1}{2} = 4032\ [m^2]$$

$$A_{separated} = 3 \times (3 \times 3.2 \times 100 + (1 + 3) \times 3.2 \times 20 \times \frac{1}{2}) = 1088[m^2]$$

The floor area of traditional toll is nearly about 4 times of separated toll. And the total length of traditional toll is about 320 m, while the separated toll is about 260 m. Consider the cost of the toll, we can find that the separated toll is much cheaper than traditional one and the throughput is nearly similiar under same allowable throughput.

6 Validating the Model Using Computer

We use program to implement both the traditional tolls plaza and the tolls we designed. Our program divided time into discrete time interval. In each interval, we calculate the throughputs to compare our design with traditional design. We set the time interval is

equal to the serving time.

The whole program contains three parts:

First, generate the data. We use program to generate the number of arriving vehicles (which obeys Poisson distribution) in each period of interval (let we say $5[s]$) and the service time for each car (which obeys exponential distribution).

Second, we use the data generate to calculate the number of vehicles which pass the tolls in each period of interval according to the Inflow model.

Third, calculate throughputs. In each small period, as we know how many vehicles will pass tolls and enter the toll plaza (let it be n_{in}), we use the outflow model to calculate how many vehicles will leave plaza (Let it be n_{out}). We also define N is the remaining vehicles in the plaza. If $n_{in} > n_{out}$, vehicles that are not able to get out will remain at plaza, this is to say, add to N. Also if $n_{in} < n_{out}$, some of the remaining vehicles at plaza will get out of it, that is to say, N will be minus.

Also, two extreme situations needed to be dealt with.

First, $n_{in} > n_{out}$ and plaza has no enough room to hold vehicles. In this situation, traffic congestion is caused and the entering speed will slow down. Equivalently, in this period of time, we can remain n_{in} unchanged but extend the time span of this period (say extend $5s$ to $10s$) and extend n_{out} accordingly. Let T' is the unit time period for this period we have

$$n_{in} + N - T'/T \cdot n_{out} = n_{max} \tag{16}$$

Therefore,

$$T' = (n_{in} + N - n_{max}) \cdot T/n_{out} \tag{17}$$

Second, $n_{out} > n_{in} + N$ and there are not enough cars. For this situation, we just move all the vehicles out. That is to say,

$$n_{out} = n_{in} + N \tag{18}$$

7 Results

7.1 Performance in Normal Condition

In normal situations, both traditional tolls plaza and tolls plaza we designed are able to remain a high performance even in high service intensity. The result is the same as our common sense that congestion seems to occur before tolls but not after them. That is because vehicles spend quite some time on tolls which limits the entering speed.

The traditional toll and separated toll are nearly overlapping in the macro graph. The line represent two conditions has the same fluctuation when receiving same input traffic flow. In the enlarged view, we can still see there exist a little difference between two conditions. The difference on the size of the toll will only affect the throughput a little

under the allowable input traffic flow.

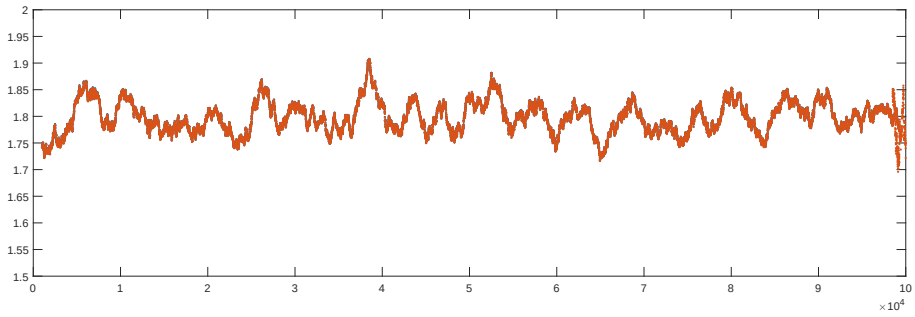

Figure 6: Normal Condition. x-axis represents time, y-axis represents throughput. Red dot represents traditional tolls. Blue dot represents separated tolls.

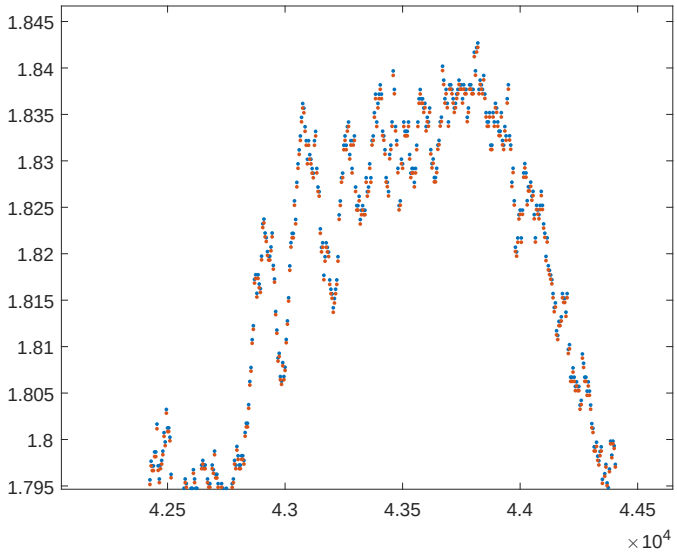

Figure 7: Normal Condition (Zoomed in). x-axis represents time, y-axis represents throughput. Red dot represents traditional tolls. Blue dot represents separated tolls.

7.2 Performance when facing accidents

However, we find that when accident occurs the reversible toll plaza designed by us perform better. Suppose a severe accident occur at 6 hour, which congest half of the roads, and 5 hours later, or at 11 hour, the tolls plaza recover from the accident.

Compared with traditional tolls plaza, plaza that designed by us can borrow a plaza (include 3 tolls and 1 lane) from the opposing direction before fully recovered. Suppose it takes 30 minutes to borrow, that is to say, from 6 hour 30 minute to 11 hour our plaza has 3 more tolls and 1 more lane than traditional ones.

The following picture shows every important time points.

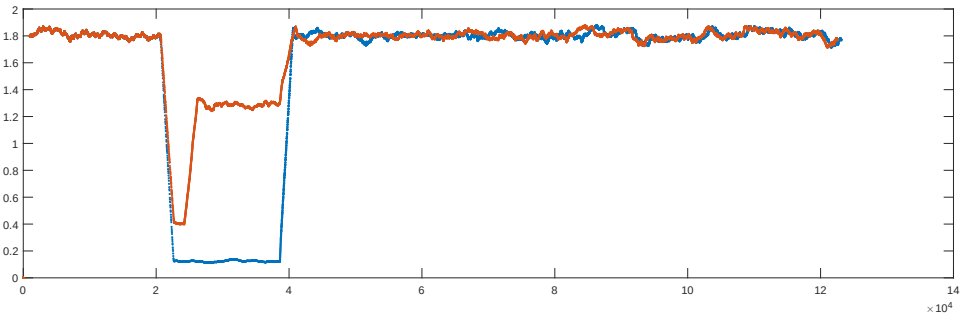

Figure 8: Accident occurs. x-axis represents time, y-axis represents throughput. Blue dot represents traditional tolls. Red dot represents reversible tolls.

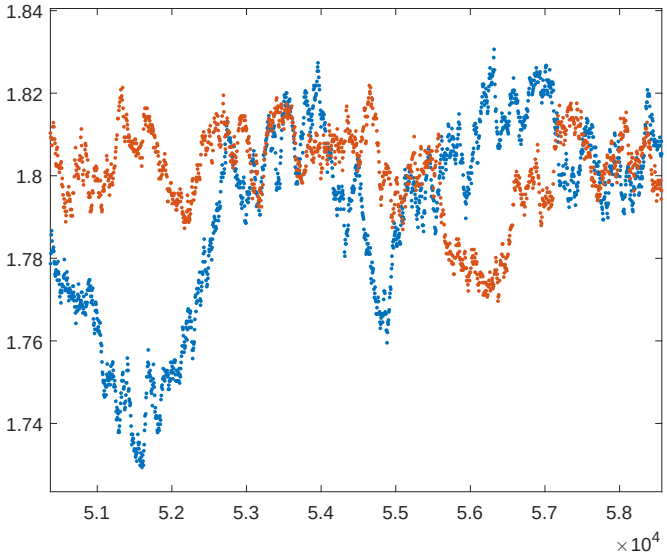

Figure 9: Accident occurs (Zoomed in). x-axis represents time, y-axis represents throughput. Blue dot represents traditional tolls. Red dot represents reversible tolls.

When accident occurred, both the throughputs our plaza and traditional plaza drop sharply. That is because both plazas are filled with vehicles and fully congested quickly.

But when our plaza has finished borrowing, its throughputs rise. That is surely because the borrowing lane. After borrowing, the throughput is lower than normal, but much higher than traditional plaza, which confirms that out plaza performs better when facing accidents.

Finally when both plazas fully recovered, the throughputs rise back to normal.

7.3 Performance when ETC and self-driving in use

According to a paper, it will achieve the best performance when the ratio of ETC and traditional tolls is $4:1$. Here, average time for passing an ETC is $3s$, average time for passing a traditional toll is $6s$. In such case, we can easily calculate average time for mixture tolls T, which is $3.4s$, so we set $T = 3.4s$.

For self-driving vehicles, they react fast and can gain faster speed than human. So we set reaction time $\Delta t = 0.1s,\ v_0 = 15m/s,\ v_2 = 15m/s$.

From the following picture, we can see both our plaza and traditional one remain high performance in high service intensity, just like normal situation. That is because even if ETC are used to achieve less time spent in passing tolls and therefore enhance passing speed, both plaza can also deal with it. We can say that unless faster tolls are invented and used, congestion seldom occurs after tolls.

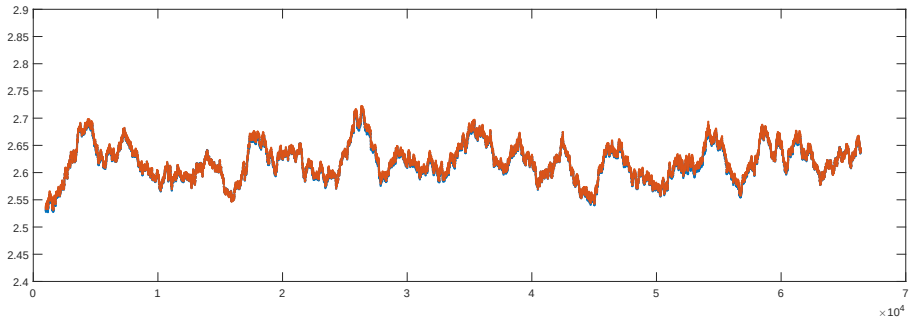

Figure 10: ETC tollbooths. x-axis represents time, y-axis represents throughput. Blue dot represents traditional tolls. Red dot represents reversible tolls.

8 Conclusions

In this problem, we developed two models: inflow model and Merging Queue model. Inflow model is used to generate the vehicles and determine how fast vehicles passing toll plazas. While Merging Queue model is used to determine how fast vehicles leaving toll plazas, in other words, throughput.

We designed two new toll plazas, reversible plaza and separated plaza. We evaluated them and traditional plaza in two aspects: area and throughput. To calculate throughput, we use inflow and merging queue models and write code to do simulation. For area, separate plaza occupies only 25% area of traditional plaza.

For throughput, both separated toll plaza, reversible toll plaza and traditional toll plaza performs well in normal situation (including light and heavy traffic), self-driving and ETC situation. But when facing accidents, reversible plaza performs much better than traditional one.

Therefore, we can conclude that the traditional toll plaza is a good design and has high

efficiency. However, the separated and reversible toll plaza performs even better in the extreme conditions or occupies smaller area. It's worthwell improving the traditional toll plaza into the two new design.

9 Strengths and Weaknesses

9.1 Strengths

- **Fully simulation of the toll**
  In this model, we considered the time consumed by charging, merging lanes, deceleration for the safety distance and waiting for the previous vehicles to exit the tolls. In addtion, the special condition of accident, ETC and self driving vehicles. Therefore, the model can fit the real conditions well.
- **Applies widely**
  Nearly every toll around the world has the same design, which is in traditional form. Our model covers all those conditions. Moreover, we considered two more new design of the toll plazas. Therefore, our model can be used widely to calculate the throughput and efficiency of the toll plazas.
- **Accessible**
  We use computer programing to help us calculate the large number of data. When we change a different condition, we only have to change the parameters of it. Therefore, if someone even doesn't understand our model, he can still use the program to help him calculate the throughput.

9.2 Weaknesses

- Our model is too sensible to the changes of the inflow of vehicles. The small changes in the waiting time at the tollbooth may affect the status of toll plaza greatly. It may because that our merging model is continuous to the changing of inflow. We can improve our model in the future.

References

[1] Y. Ni, Z. Dong Analysis and Simulation of the Features of Queues in Highway Toll Station, 2016

[2] Specification for Design of Expressway Toll Station and Toll Plazas, 2003

[3] Sundarapandian, V. (2009). "7. Queueing Theory". Probability, Statistics and Queueing Theory. PHI Learning.

[4] `http://www.state.nj.us/transportation/about/safety/`

[5] `http://www.statemaster.com/graph/trn_tol_roa_mil-transportation-toll-road-mileage`

Appendices

Appendix A Matlab Codes

Here is the matlab program we used in our model as follow. This program is used for generating the incoming vehicles using Poisson distribution.

Input matlab source:

```
function poisson(ex,n) %%the input parameter 'ex' is rhe mean of output data, and 'n' is the sm
string1 = ['poisson_' num2str(ex) '_' num2str(n) '.txt'];
fid11 = fopen(string1, 'wt');
N = random('poisson',ex,1,n);
for i = 1:n
fprintf(fid11,'%g \r\n',N(i));
end
fclose(fid11);
m  = sum(N);
M = exprnd(1,1,m);
string2 = ['poisson_' num2str(ex) '_' num2str(n) '_exprnd' '.txt'];
fid2 = fopen(string2, 'wt');
for i = 1:m
fprintf(fid2,'%f \r\n',M(i));
end
fclose(fid2);
end
```

```
w = [];

for L = 3:11
 D = 3.2; B=6;l = 15;l0 =100;a = 4;d0 = 1;dt = 0.5;v0=10 ;v2=10 ;a1 =2;a2=2;
 v = 0;
 v1 = v0;
 while  abs(v-v1)>0.001
 v = v1;
d = d0 + v1* dt;
td = (a+d)*(a1+a2)/(a1*v2)+sqrt(2*a2*(a+d)*(D^2+L^2))/v2;
N = (B*D*l0+(B+L)*D*l*1/2)/(D*(a+d));
s0 = 0;i=0;
while i <  B/2
    s0 = s0 + sqrt((B-2*i-1)^2*D^2/4 +l^2)/B;
    i=i+1;
end
t0 =(s0+l0)/v0;

n = floor(N*(B-L)/L)+1;
t1 = (n*td  + t0*N)/N;

v1 = (s0+l0)/t1;
end
 W  = v1/(a+d) *L;
 w = [w W];

end
```

Appendix B C++ Codes

Here is the C++ program we used in our model as follow. This program is used for calculate the throughput.

Input C++ source:

```
// MCM.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <cmath>
#include <vector>
using namespace std;
ifstream fin("F:\\res.txt", iostream::in);
ofstream fout("F:\\throughputs.txt", iostream::out);
vector<pair<double, double>> w;
bool flag = false;
const double u0 = 3.4;
const double D = 3.2;
const double l = 60;
const double l0 = 100;
const double a = 4;
double d = 1;
const double dt = 0.1;
const double d0 = 1;
const double v0 = 15;
const double v2 = 15;
const double a1 = 2;
const double a2 = 2;
double v1 = v0;

class plaza
{
public:
    double B;
    double L;
    double N;
    double u;
    double maxN;
    plaza(int nB,int nL)
    {
        u = 1;
        N = 0;
        B = nB;
        L = nL;
        //maxN = 3 * (3 * 3.2 * 30 + (3 + 1)*3.2 * 20 / 2) / (3.2 * 5);
        maxN = (B*D*l0 + (B + L)*D*l/ 2) / (D*(a + d));
    }
    double GetOutNum()
    {
        /*if (N <= 0.8*maxN)
            return v0 / (a + d);*/

        d = d0 + v1*dt;
        double td = (a + d)*(a1 + a2) / (a1*v2) + sqrt(2 * a2*(a + d)*(D*D+ L*L)) / v2;
        double s0 = 0;
```

```
        int i = 0;
        while (i < B / 2)
        {
            s0 = s0 + sqrt((B - 2 * i - 1) *(B - 2 * i - 1) * D*D / 4 + l*l) / B;
            i = i + 1;
        }
        double t0 = (s0 + l0) / v0;
        int n = int(N*(B - L) / L + 1);//šćţÀt'ÎÊý
        double t1 = (n*td + t0*maxN) / maxN;
        v1 = (s0 + l0) / t1;
        double w = v1 / (a + d)*L;//*N/maxN;
        //std::cout <<"w:"<< w << endl;
        return w*u0;
    }
    bool InandOut(int x)
    {
        if (x < 0)
        {
            N +=x;
            if (N < 0)
                N = 0;
            return true;
        }
        if (N + x <= maxN)
            N += x;
        else
            return false;
        return true;
    }
};
int GetIn()
{
    if (flag)
        return 0;
    int tmp;
    fin >> tmp;
    return tmp;
}
void exchange(plaza*a)
{
    double tot = a[0].N+a[1].N+a[2].N;
    tot /= 3;
    a[0].N = a[1].N = a[2].N = tot;
}
void Init(plaza*a)
{
    a[0].B =9;
    a[0].L =3;
}
int plazanum;
int main()
{
    plaza *a;
    a = new plaza(9,3);
    Init(a);
    int tmp;
    int it = 0;
    double tottime=0;
    //bool flagoccur = false;
    //bool flagsolve = false;
    //bool flagborrow = false;
```

```
    while (!((a->N == 0)&&flag))
    {

        if (it%100==0)
            cout << it<<" ", cout << a->N << " " << flag << endl;
        it++;
        //exchange(a);
        /*for (int i = 0; i < plazanum; i++)
        {
            tmp = GetIn();
            a[i].InandOut(tmp - a[i);
        }*/
        int in = GetIn();
        if (in == -1)
        {
            flag = true;
            in = 0;
        }
        double out = a->GetOutNum();
        if (out> (a->N+in))
            w.push_back(pair<double, double>(a->N+in, u0));
        else
            w.push_back(pair<double, double>(out, u0));
        if (!a->InandOut(in-out))
        {
            a->u = (a->N + in - a->maxN) / out;
            a->N = a->maxN;
            w.back().second = a->u*u0;
        }
        tottime += w.back().second;

    }
    double outresult = 0;
    fout << "result\n";
    for (const auto& elem : w)
    {
        fout << elem.first << " " << elem.second << endl;
        outresult += elem.first*elem.second;
    }
    std::system("pause");
    return 0;
}
```

```
// DATA.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <queue>
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
class Desk
{
public:
    double time;
    int num;
    Desk() { time = 0; num = 0; }
    Desk(double newtime, int newnum)
    {
```

```
        time = newtime;
        num = newnum;
    }
    bool operator<(const Desk&otr)const
    {
        if (time != otr.time)
            return time > otr.time;
        return num > otr.num;
    }
};
double curtime = 0;
int desknum = 9;
queue<double> cartime;
queue<double> cararrive;
priority_queue<Desk> thing;

int main()
{
    ifstream fcar("F:\\car.txt", ifstream::in);
    ifstream ftime("F:\\time.txt", ifstream::in);
    ofstream fout("F:\\res.txt", ofstream::out);
    double tmp;
    int tmpnum;
    while (fcar >> tmpnum)
    {
        for (int i = 0; i < tmpnum; i++)
        {
            ftime >> tmp;
            cararrive.push(curtime);
            cartime.push(tmp);
        }
        curtime++;
    }
    Desk tmpthing;
    for (int i = 0; i < desknum; i++)
    {
        thing.push(tmpthing);
        tmpthing.num++;
    }
    curtime = 0;
    int totcalc = 0;
    while (!thing.empty())
    {
        int calc = 0;
        while ((!thing.empty()) && (thing.top().time < curtime + 1))
        {
            tmpthing = thing.top();
            thing.pop();
            if (tmpthing.time == 0)
                calc--;
            if (!cararrive.empty())
            {
                tmpthing.time = max(tmpthing.time, cararrive.front()) + cartime.front();
                cararrive.pop();
                cartime.pop();
                thing.push(tmpthing);
            }
            calc++;

        }
        totcalc += calc;
```

```
        fout << calc << '\n';
        curtime += 1;
    }
    cout << totcalc;
    system("pause");
}
```