# 1 A brief description of the algorithm

## 1.1 basic assumption and parameters

The platform considers the maximum acceleration of the vehicle is $a_{max}$, which is generally used in the acceleration process and tbe braking process. Also, the maximum speed limit is set to $v_{max}$, while the minimum speed limit is set to $v_{min}$.

No matter which strategy is chosen, if the acceleration given by the strategy is greater than the maximum acceleration $a_{max}$, or the calculated speed is greater than the maximum speed or less than the minimum speed, the program will automatically set these values to the boundary value. When the vehicle has to stop because of the red light, the minimum speed of the vehicle is set to 0, describing the parking behavior which may exist, however, the behavior of reversing is never allowed.

Also, we assume the expected speed of the vehicle is set to 80% of the maximum speed limit.

In order to simpliy the simulation process, we assume that during the process of driving, there is no lane changing behavior. This hypothesis is also acceptable in the real life, because of the restrictions imposed by driving habits and traffic laws & regulations, the vehicle should finish changing lane before enter the traffic junction. Therefore this assumption is reasonable. What's more, just from the perspective of traffic efficiency, lane-changing can only reduce the average efficiency of the traffic, therefore, as the primary inspection of the platform is efficiency, lane-changing situation is not taken into consideration.

Therefore, the car driving in the lane can be described as a queue, e.g. $Q_{in}$

## 1.2 The Generation of the vehicles

It is assumed that the arrival process of the vehicle is approximately a Poisson process, which is limited and adjusted by the following algorithm

Assuming that the intensity of the Poisson process is$\lambda$, the arrival time interval of the vehicle $S_n, S_{n-1}$can be expressed as

$$P((S_n - S_{n-1} \leq t) = 1 - e^{-\lambda t}$$

Because the simulation time scale is set to 0.1s, the average hourly traffic flow is

$$\bar{F} = 36000\lambda$$

For the traffic flow in one direction, after the average hourly traffic flow is set by the traffic flow adjustment slider described above, The vehicles in this direction will be generated according to the Poisson process with a specified intensity of $\lambda = \dfrac{\bar{F}}{3600}$, and will be pushed into the pending queue $Q_{pending}$in each direction.

After generating the vehicle, the next step is to consider which lane the vehicle belongs to, and it is assumed that the possibility of the vehicle appearing in each safe lane is same. Firstly, the condition to determine a safe lane is that the distance of last vehicle in the lane and line which the vehicles are generated ($S_{start}$) is greater than the safe distance $S_{safe}$, that is,

$$X_{-1} - S_{start} \leq S_{safe}$$

Screening all lanes that meet the situation above in this direction, and place the vehicle in $Q_{pending}$ into one of these lanes randomly, while the position of the vehicle newly generated is set to $S_{start}$.

If there is no lanes suitable for placing the vehicle currently, the vehicles will remain in $Q_{pending}$ and wait for the next simulation session.

The generation of the vehicle is done in the second step of the simulation cycle, just following the process control model.

## 1.3 The formation and evacuation of the queue in the intersection

To describe the phenomenon of queuing aat the intersection, a queue $Q_{block}$ is introduced to each lane towards the intersection. And according to the current signal phase and the length of the queue, all of the control strategies mentioned take these simple algorithm below.

1. Red Light

   If the distance between the first vehicle in $Q_{in}$ and the last vehicle in $Q_{block}$ is less than the control range $S_{control}$, the first vehicle in $Q_{in}$ brake and is expected to stop just behind the last vehicle in $Q_{block}$. The expected distance between these two vehicles is defined as $S_{stop}$, describing the distance between the vehicles in the $Q_{block}$.

   If the $Q_{block}$ is empty and the distance between the first vehicle in $Q_{in}$ and the stopping line is less than $S_{control}$, the first vehicle in $Q_{in}$ brake and is expected to stop just on the stopping line$S_{end}$.

   If the first vehicle in the $Q_{in}$is too far that neither of the two above are satisfied, the first vehicle in the $Q_{in}$ is supposed to drive freely.

   Also, the vehicles in the $Q_{block}$ should stop and wait until the light truns green.

2. Green Light distance between the first vehicle in $Q_{in}$ and the last vehicle in $Q_{block}$ (if it exists) should not be less than $S_{safe}$. And a braking is taken if the distance mentioned above is too short.

Therefore, the brief idea of the algorithm above can be described as the fake code below

```
if (Q_block.empty()){
        if (Light == Green)
                Q_in.first().drive_freely();
        else
                Q_in.first().brake_to(S_end);
}
else{
        if(Q_block.last().pos-Q_in.first().pos< S_control)
                Q_in.first().brake_to
                (Q_block.last().pos-S_stop);
        else
                Q_in.first().drive_freely();
}
```

Where the brake_to(desired_pos) method means braking to a desired position.

Meanwhile, the growth and the dissipation of the queue $Q_block$ follow the method below.

1. Increase

   If the first vehicle in $Q_{in}$ is closer than $S_{stop}$ from the last vehicle in $Q_{block}$ (it is set to $S_{end}$ if the $Q_{block}$ is empty), it is removed from the $Q_{in}$ and pushed into the $Q_{block}$

2. Dissipation

   When the traffic light is green, all of the vehicles in the $Q_{block}$ is moving forward at the speed of $v_{dis}$, since the speed is relatively low, the acceleration process and any other phenomenon can be ignored. When the vehicle moves over the stopping lane and enters the intersection, it is removed from the $Q_{block}$

And the method above can be described as the fake code below

```
if (light == green){
        Q_block.all_move_forward(v_dis)
}
else{
        if(!Q_block.empty()){
                if(Q_block.last().pos-Q_in.first().pos
                        < S_stop){
                        Vehicle v;
                        v=Q_in.getfirst();
                        Q_block.push(v);
                }
        }
        else{
                if(S_end-Q_in.first().pos< S_stop){
                        Vehicle v;
                        v=Q_in.getfirst();
                        Q_block.push(v);
                }
        }

}
```

All above describe the formation and evacuation of the queue in the intersection and its interaction with other models. The differences caused by the assumption can be ignored. Thanks to this model, it is easy for us to decouple the driving model and the queuing model, which makes it much easier to implement the other algorithm

## 1.4 The implementation of the car-following method