Yellow Lights & Multiple Velocity Functions: Measuring Traffic Flow Efficiency Based on Flux

Eddie Shim
Edd Kim
Ryan Yue

I. Introduction

In this project, we decided to expand and explore upon the original traffic model that was taught in class. Considering our possibilities for this project, we thought traffic simulation would be the most interesting to model in order to test some tangible real life scenarios we could think of and see if our intuition matched our mathematical models.

Some interesting scenarios we believed would be interesting to model involved efficiency of traffic flow based on different conditions and parameters imposed on the cars. After discussing a few different project ideas, we decided to pursue two main ideas we believed would have the most interesting results.

The first part of our project revolves around traffic flow dependent on different street light conditions. We augmented the function setlight(t) in the initial traffic model in order to 1) observe how traffic flow changes when we add in a yellow light and 2) observe how traffic flow changes when we change the timing of the lights from simultaneous to non-simultaneous, periodic times.

In order to simulate the yellow lights and the periodic delay per light, we added a new function getLight(t, b) which takes parameters of the current time step and index of the current traffic light, then returns the index corresponding to green, yellow, or red. To calculate the delay, a global parameter representing delay was invoked and multiplied by the traffic light index *b*. The result was subtracted by the delay value (since values in MATLAB start with 1 instead of 0), and the value was assigned to the time step *t*. TBC...

First off, we decided to define how we would measure efficiency of traffic flow. Keeping a few parameters fixed (tmax = 3000, dt = 2, dmin = 15, dmax = 30, vmax = 50, blocks = 4) we decided to keep a count of how many cars exit the system at which dt time interval. As an output, we have a graph with number of cars exited on the y-axis, and time interval on the x-axis. Furthermore, we defined the base case as the simulation with 0 lights and cars simply streaming through. Every other case would measure efficiency relative to this base case as a percentage (number of cars exited / base case number of cars exited).

When within a distance d<dmin close enough to the light, cars would stop at red, maintain their velocity at green, and for yellow, treat the light as if it were a red light at a fixed distance ahead of the light itself. In this case, if the car crossed the yellow light before it turned red, it would slow its velocity as if the light were red at a distance ahead. If the light turned red

before the car crossed the yellow light, the car would stop at the light. The specifics of the model code are detailed below in the 2nd and 3rd section of this write up.

Furthermore, we decided to experiment on light timings. By having the streetlights be nonperiodic, we were able to keep a more constant flow of traffic moving through the street. Thus, we hypothesized that there could be an optimal street light periodic timing which would keep traffic flow as efficient as possible. This effect is visible in New York City traffic as well, where streets are short in length and cars are dense but lights change periodically so cars are able to move through multiple blocks with ease.

For the second part of our project, we decided to augment the velocity function, v(d). This function takes in a car's distance to the next car or light as a parameter and sets the velocity of the car. We created to separate functions, v1(d), and v2(d) in order to have different sets of cars moving at different velocity functions. By doing this, we were attempting to have the simulation be closer to real life in that the model would account for some drivers who prefer slower driving and some drivers preferring faster driving.

II. Model Discussion

We first started modeling the simulation on a one-dimensional grid representing a street, with cars moving from left to right on the street. Global parameters used across multiple files included *dmin* (the minimum distance in meters allowed between each car), *dmax* (the maximum distance allowed between each car), *tgreen* (the total time in seconds a traffic light stays green), *tyellow* (the total time a traffic light stays yellow), *tred* (the total time a traffic light stays red), *delay* (the sequential delay between each traffic light changing colors), *L* (the length of a block), and *nblocks* (the total number of blocks / traffic lights).

The iteration begins by looping through *nblocks* traffic lights to retrieve their colors given the current time step.

We also extended the traffic simulation provided to us in recitation to include a yellow light segment as well as a sequential delay between each traffic light changing colors.

III. Matlab Code

```
%Simualtion of traffic flow
%v1,v2
%Eddie Shim, Edd Kim, Ryan Yue
%5/2/16

clear
close all
```

```
% initialize global parameters
global dmin dmax vmax  nblocks tgreen tred  L dmin2 dmax2;
dmin=15;                      % m
dmax=30;                      % m
vmax=50;

tgreen=120;                   % s
tred=120;                     % s
L=200;                  % m
nblocks=4;

dmin2 = 30;
dmax2 = 60;

nblocks=4;
numCars= 0;
ArrayCars = [0];
count = 0;

p=rand(1,1000);

% set positions of cars at initial time
x=zeros(1,1000);
firstcar=[1 1 1 1];
lastcar=[0 0 0 0];
% time parameters
tmax=3000;                    % s
dt=2;                   % s
clockmax=ceil(tmax/dt);

% variables to record
tsave=zeros(1,clockmax);
ncw=zeros(1,clockmax);        % number of cars waiting to enter the road

% rate of cars coming into the road
R=1/5;                        % 1/s

figure
% set colors
grey=[0.4 0.4 0.4];
green=[0 1 0];
red=[1 0 0];
col=zeros(nblocks,3);
% set position of the figure
set(gcf, 'Position', [250  250  1000  400]);
% draw a line which represents the road
line([-50 800],[0 0],'color',grey,'linewidth',2);
xlim([-50 800]);
```

```
ylim([-0.4 1]);
axis off
hold on
% set the three traffic light positions
for i=1:nblocks
      line([i*L i*L],[0 0.2],'color',grey,'linewidth',2);
end
% put cars on road
h=plot(unique(x),zeros(1,length(unique(x))),'ok','markerfacecolor','k','marker
edgecolor','k');

for clock=1:clockmax
      t=clock*dt;
      tsave(clock)=t;
      % set the traffic light
      s=setlight(t);                              % define setlight in a
seperate function file

      % set the light colors and draw the lights
      for i=1:nblocks
      col(i,:)=s(i)*green+(1-s(i))*red;

plot(i*L,0.2,'color',col(i,:),'marker','o','markerfacecolor',col(i,:),'markers
ize',16)
      text(i*L-2,0.2, int2str(i));
      end


      % calculate random arrival of cars
      if (rand<R*dt)                                % IMPORTANT: R*dt should be
less than 1
      lastcar(1)=lastcar(1)+1;
      tenter(lastcar(1))=t;                % record enter time
      end

      % move the cars by blocks if the block is not empty
      for b=nblocks:(-1):1                        % start from the most ahead
block
      if (firstcar(b)<=lastcar(b))         % if the block is empty, no cars to
move
            c=firstcar(b);
            if (s(b))                             % green light
            if (c==1) || (b==nblocks)      % no cars ahead
                  d=L;
                  %d=x(c-1)-x(c);
            else
                  d=x(c-1)-x(c);
            end
            else                                % red light
            d=b*L-x(c);                   % distance to light
            end

            if (p(c)<0)
            x(c)=x(c)+dt*v1(d);                 % define v in a seperate
function file
            else
            x(c)=x(c)+dt*v2(d);
            end
            if (x(c)>b*L)                       % 1st car passed the light
            firstcar(b)=firstcar(b)+1;
            if (b<nblocks)
                  lastcar(b+1)=lastcar(b+1)+1;
            else                          % reach destination
                  texit(c)=t;             % record exit time
                  numCars = numCars+1;
            end
            end
            c=c+1;
            while (c<=lastcar(b))               % update positions of the rest
cars

            if (p(c)<0)
                    x(c)=x(c)+dt*v1(x(c-1)-x(c));
            else
                    x(c)=x(c)+dt*v2(x(c-1)-x(c));
            end
            c=c+1;
            end
      end

      end

      ArrayCars = [ArrayCars numCars];
      % update positions of cars on the figure
      set(h,'xdata',unique(x),'ydata',zeros(1,length(unique(x))));
      pause(0.1)
end

disp(numCars);
figure(2)
t = 1:clockmax+1;
plot(t, ArrayCars, 'linewidth', 1.5);
```

```
function [v] = v1(d)
global dmin dmax vmax;
```

```
if (d<=dmin)
   v = 0;
else
   v = (log(d/dmin)/log(dmax/dmin));
end
```

```
function [v] = v2(d)
global dmin dmax vmax;

if (d<=dmin)
   v = 0;
else
   v = 5*(log(d/dmin)/log(dmax/dmin));
end
```

```
function [s] = setlight(t)

global nblocks tgreen tred;

tt = mod(t, tgreen + tred);
if (tt < tgreen)
      s = ones(1, nblocks);
else
      s = zeros(1, nblocks);
end
End
```

IV. Results and Discussion

When we measured efficiency, we used the case where there were 0 lights as the base case. We found that the number of cars that exited the system was 104, and we used this as 100% efficient. To calculate efficiency of the other scenarios, we used the number of cars that exited the current system divided by 104.

With 1 light, we found that the number of cars that exited the system was 102, and it ran at 98.1% efficiency (102/104). We were very surprised because the efficiency was very close to 100%, only 1.9% off. We expected a much worse efficient simulation and eventually concluded that because there was only 1 light, the efficiency was based on the average velocity of the system. With 0 lights, the cars would move slower at the beginning of the system and gradually speed up towards the end. With 1 light, the cars would do the same unless at a red, then they would slow down and stop. When the light turned green, multiple cars would move faster than the velocity of the cars in the 0 light case. With the constant slowing and speeding of the cars in the 1 light case, the average velocity ended up to be very close with the average velocity in the 0 light case, which is shown by the efficiencies of the two cases.

The 2 light case was also very surprising. The efficiency between 0 lights and 1 light was only a 1.9% difference, so we expected no more than a 10% difference between 1 light and 2 lights. However, we were shocked when we found that the number of cars that passed through 2 lights was only 68, which meant only 65.4% (68/104) efficiency. 2 lights was much slower than 1 light and was about 33% less efficient.

The results of the 3 and 4 light cases were very similar to the 2 light case. For the 3 light case, there were 65 cars that exited a system, resulting in 62.5% (65/104) efficiency. For the 4 light scenario, there were 62 cars that exited the system, resulting in a 59.6% (62/104) efficiency. Comparing the 2, 3, and 4 light cases, not much changed. The efficiency only went down by about 3% every time a new light was added. Before the simulation, we did not know what to expect. The change in efficiency from the 0 to 1 light cases was so marginal, and the change in efficiency from the 1 to 2 light cases was extreme. We can now assume that if we were to add more traffic lights, the efficiency would be lowered at a constant or near constant rate.

In the next part of our project, we introduced a velocity v2 that was 5 times the original velocity v1. For the base case of v1, we used the 4 light scenario in the previous section (104 cars exited the system). For the base case of v2, we used the 4 light scenario and set all the cars' velocities to v2. The number of cars that exited the system was 173. We used this number to calculate efficiency relative to each velocity. When we introduced both velocities into the system, we set 70% of the cars to have velocity v1 and 30% of cars to have velocity v2 (all randomly chosen). This was to simulate the real world where there are some faster drivers and some slower drivers. When we ran the simulation, we found that 64 cars exited the system. We then compared them to the cases where everyone drove fast (v2) and when everyone drove at the regular speed (v1). We found that in relation to v1, the efficiency was 61.5% (64/104). In relation to v2, the efficiency was only 37% (64/173). This shows us that it is better to be a slower driver because it is more efficient. If you are a driver that likes to drive faster than the speed limit, it will not be very efficient because you will most likely be surrounded by other drivers who obey the speed limit and will thus cause you to slow down.

In the final part of our project we incorporated a yellow light into our simulation. We modified the velocity function so that the cars would either speed up or slow down at a yellow light (similar to the real world). There were two parts to this as well. The first case we simulated all the traffic lights to change simultaneously. We found that 61 cars exited the system, giving a 58.7% (61/104) efficiency rate. The second case we simulated all the traffic lights to change at a short delay, similar to the real world. We found that 60 cars exited the system, giving a 57.7% (60/104) efficiency rate. We were expecting the second case to be more efficient than the first case, especially since the second case was more similar to the real world (another student's project ultimately resulted in him being able to drive 50+ streets without stopping, implying the delay of the change in lights). We concluded that the reason the second case was slightly less efficient was the velocity. If we had chosen different velocities for the cars so that the average velocity of the system increased, then the number of cars that exited the system would have gone up and thus increase efficiency.