Johan Leijonhufvud

Final Project

Multilane highway simulation

Introduction

I have in this project tried to get an accurate simulation for the flow of cars on a 2 lane and 3 lane highway. I have tried to make my model for this close to a real world car flow. First I will below explain how I have set up my model for simulating a real world car flow and then I will try to explain in detail why I have made these choices. After that I will go on to present what I have analysed, why I have thought that would be interesting to analyse, my results of the analysis with statistical error and at last conclusion.

Problem formulation

Rules:

1. The position of a car is periodic with the period roadlength.

2. If $v_i < v_{max}$, increase the velocity v_i of car i by one unit, that is, $v_i \rightarrow v_i + 1$. Every car has its own maximum speed, v_{max}.

3. Calculate the distance to the next car in the same lane (if it exists) and the distance/distances to the next car in the adjacent lane/lanes. A car in lane 1 only has to calculate the distance to the next car in lane 1 and 2. A car in lane 3 calculates the distance to the next car in lane 3 and 2. A car in lane 2 calculates the distance to the next car in lane 2 and the distances to the next cars in the two adjacent lanes.
   Say for the case of car i in lane 2, d_1, d_2, d_3 are the distances to the next cars in the different lanes (it works analogously for cars in the other two lanes but then only d_n and d_{n+1}need to be calculated). If d_k =max(d_1, d_2, d_3)$\leq v_i$ then v_i becomes $d_k - 1$ and car i switches to lane k, or if k is its current lane then it stays there.
   If $d_m = d_k$ =max(d_1, d_2, d_3)and $m \neq k$, then pick lane m or lane k at random and switch to that. If there would be no cars in front of car i in its current lane or in any of the adjacent lanes then switch to that lane and don't change the car's velocity. If there are multiple (unlikely) pick one at random. Just to clarify: if a car switches lanes, that can only be to an adjacent lane.

4. With probability p, reduce the velocity of a moving car by one unit: $v_i \rightarrow v_i - 1$, only do this when $v_i > 0$ to avoid negative velocities. p is a constant set to 50%.

Johan Leijonhufvud

Parameters:
The number of lanes (a number that is either 1, 2 or 3).
The road length
The number of cars
v_{max} for every car

Example:
Just to give the reader a better understanding of what i mean when I say "distance to the next car" I will give an example. Say we have two lanes and two cars and a road length of 4. Car 1 is in lane 1 and position 2, car 2 is in lane 2 and position 1. We calculate the distance from car 1 to the next car in lane 2. Since car 1 has position 2 and car 2 has position 1 one may think this cannot be done. But you can since we have defined the position of the cars to be periodic, meaning we can represent the road as a circular path. The distance car 2 is in front of car 1 is:
(car 2 position - car 1 position + roadlength) = 3. We can also calculate the distance car 1 is in front of car 2 and that is just: car 1 position - car 2 position = 1.

Motivating choices:
Here I will explain why I have set up the rules for this simulation the way that I have. I will go over every point under the subheading "Rules" one at a time.

1. The reason why the position of a car is periodic is because that represents real world car flow better compared to if car position was not periodic. If position was not periodic then the cars with higher v_{max} would after a long time be far ahead of cars with lower v_{max} and there would be almost no traffic jams. That would not be a very good representation of real world traffic. In reality there will always be cars in front.

2. This change models the process of acceleration to maximum speed. We want the traffic flow to be as good as possible with the given parameters and you should not be able to accelerate more than one per timestep for it to be realistic.

3. Here I am only saying that the driver will make rational decisions. Switch to the lane that has the longest distance to the next car in front. If there is a lane with no cars, switch to that lane. If there are no other cars in the current lane, stay in the lane.

4. This is just to add some randomization to the simulation.

Analysis

We will analyse how the flowrate, defined as $\frac{\sum_{i=1}^{n}(v_i)}{road\ length}$, where n is the number of cars, is depending on car density, defined as $\frac{number\ of\ cars}{road\ length}$, for a one lane highway first, then for a two lane highway and at last for a three lane highway. Since we are investigating the flow of cars on a highway we set half of the cars v_{max} to 10 and the other half to 12, for the two lane case. The reason is because 10 could represent 100km/h and 12 could represent 120km/h. That is around the speed most cars are going on a highway. The reason why I have two different v_{max} for the two lane case is because it could in reality represent that there are basically two speeds on a two lane highway. Since in reality, cars who are going in the right lane will generally be going a bit slower compared to cars in the left lane. This will not be considered here but it gives the reason why I have chosen two different values for v_{max}.

For the one lane case I will set $v_{max} = 10$ for half of the cars and $v_{max} = 12$ for the other half and compare this to if I had set all the cars v_{max} to 10. Since cars can't pass each other my hypothesis is that there won't be much difference in flowrate for the two setups at a stationary state.

For the three lane case we set v_{max} to 10 for one third of the cars, one third to 11 and one third to 12. This makes sense for the same reason the setup for the two lane case made sense.

The road length is chosen to 100 for all cases, it could represent ~278m if every timestep was one second. This is because:
$\frac{road\ length}{v_{max}} = \frac{real\ road\ length}{v_{real\ max}}$.

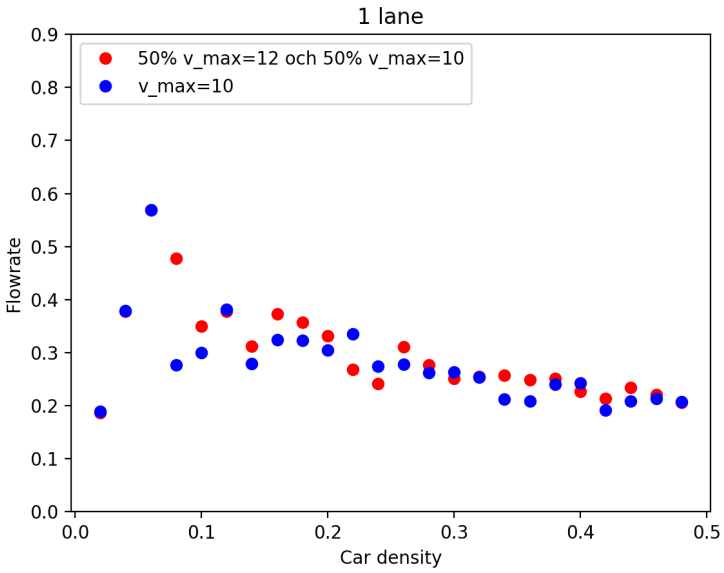
I wrote before that $v_{max} = 10$ could correspond to the real velocity 100km/h and we set the road length to 100 and that gives us the real road length which is: $real\ road\ length = \frac{road\ length}{v_{max}} \cdot v_{real\ max} = \frac{100}{10} \cdot \frac{100}{3{,}6}\ m \approx 278\ m.$ The road length should not matter, if I choose the road length to be twice as big, then the flowrate should be the highest the same car density as before (twice as many cars). I will show this in a plot for the three lane case after I have investigated the one lane, two lane and three lane cases for road length 100.

The number of cars is a variable that is $1 < number\ of\ cars \leq road\ length$ for the one lane case,$1 < number\ of\ cars \leq 2 \cdot road\ length$for the two lane case and $2 < number\ of\ cars \leq 3 \cdot road\ length$ for the three lane case. We add two cars in a simulation at a time for the two car and one car cases. One with $v_{max} = 10$ and one with $v_{max} = 12$. For the one car case we also run another simulation where we add 2 cars at a time but now both have $v_{max} = 10$. This is because we want (in the one lane case) to simulate both setups with the same car densities and then it will be easier to compare at those points. We add three cars in a simulation at a time for the three car case one with $v_{max} = 10$, one with $v_{max} = 11$ and one with $v_{max} = 12$.

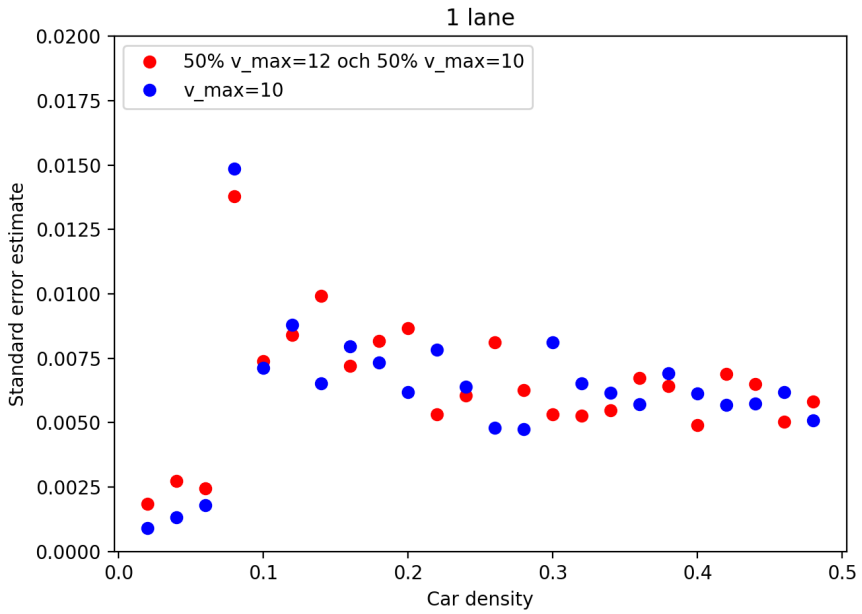
For the one lane case I have put the initial positions so that each car is a distance 1 apart. For the case where half of the cars have $v_{max} = 10$ and the other half have $v_{max} = 12$ then I have placed every other car in the lane to have $v_{max} = 10$ and every other $v_{max} = 12$. For the multilane cases the initial positions of the cars are chosen to have initial positions that are one apart and every other or every third car in a specific lane has $v_{max} = 10$, $v_{max} = 11$(3 lanes) and $v_{max} = 12$. For example, every second/third (2 lanes or 3 lanes) car in lane one has $v_{max} = 10$, every second/third car in lane two is also $v_{max} = 10$ and $v_{max} = 12$, but at different positions. The same goes for cars with $v_{max} = 11$in the three lane case. This is just because I want the flowrate to become stationary as fast as possible, otherwise the initial positions and the lanes of the cars does not really matter.

Johan Leijonhufvud

1 lane:

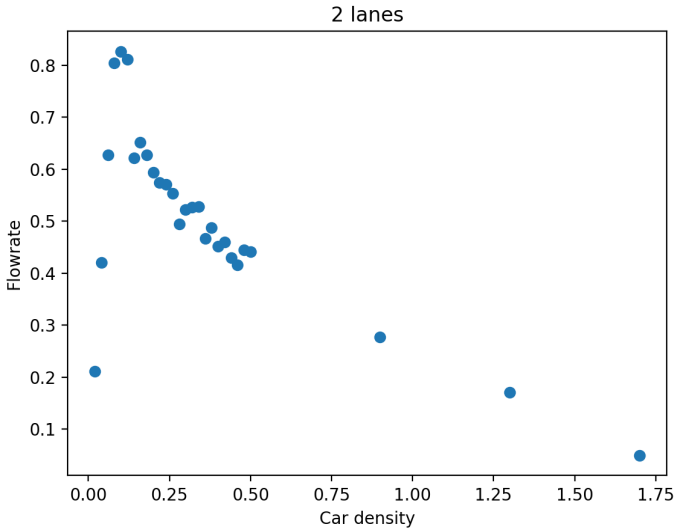


I ran the simulation above for 300 timesteps for each car density (represented on the x axis). I measured the flowrate at every timestep from 240 to 299, what you see on the y axis in the plot above is the average of those points flowrate. The maximum flowrate above is when car density is 0,06 (6 cars). The flowrate for the two setups are very similar, just as expected. Below is the statistical error measured over the time steps from 240 to 299



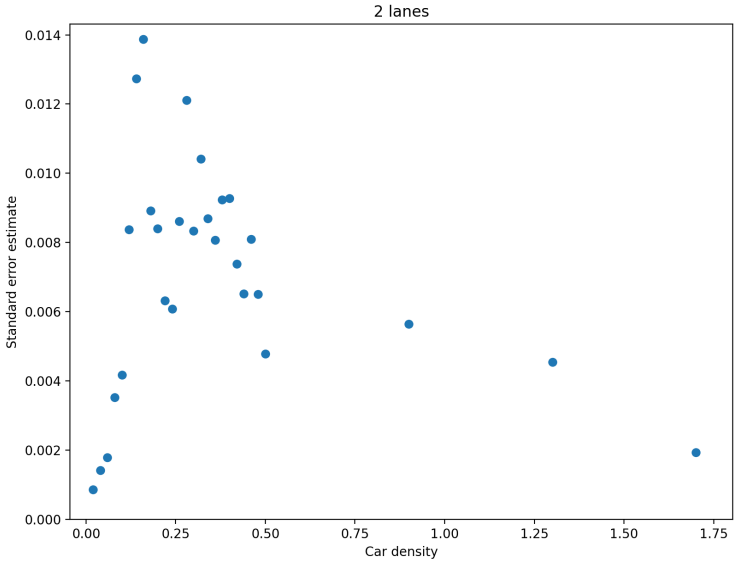
As one can see, the error is pretty low and we can with high probability say that a car density of 0,06 (standard error ~0,0025) gives the best flow rate.

2 lanes:



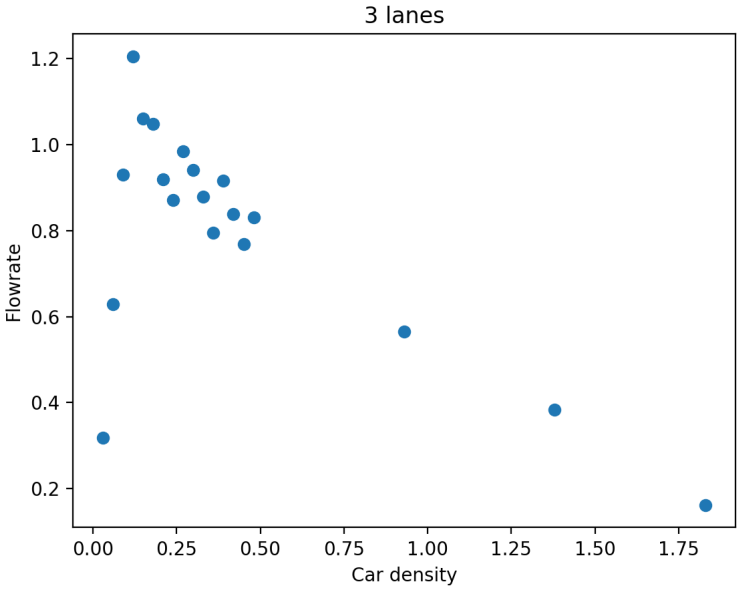
I ran the simulation above for 300 timesteps for each car density (represented on the x axis). I measured the flowrate at every timestep from 240 to 299, what you see on the y axis in the plot above is the average of those points flowrate. The maximum flowrate above is when car density is 0,10 (10 cars). Below is the statistical error measured over the time steps from 240 to 299

Johan Leijonhufvud



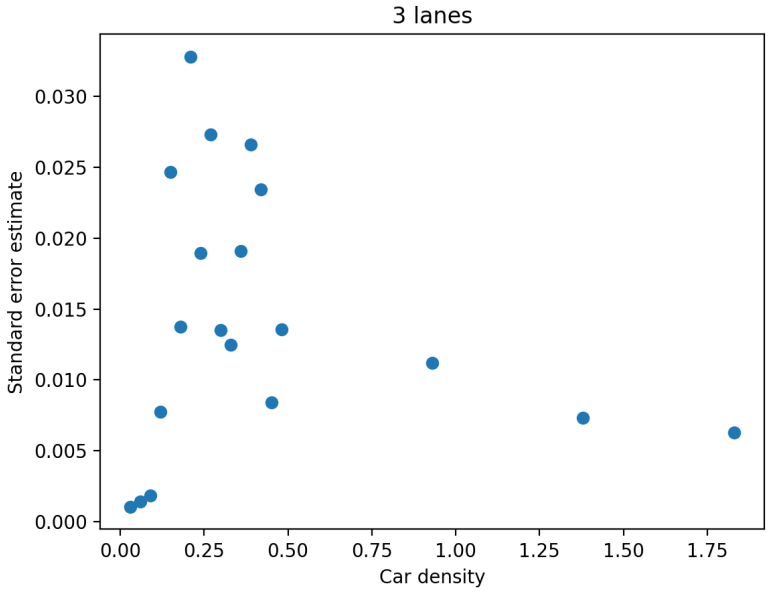
Car density 0,08 and 0,12 have a flowrate close to car density 0,10, but that only strengthens the argument that car density 0.10 gives the biggest flowrate. As one can see, the error is pretty low and we can with high probability say that a car density of 0,10 (standard error ~0,004) gives the best flowrate.

3 lanes:



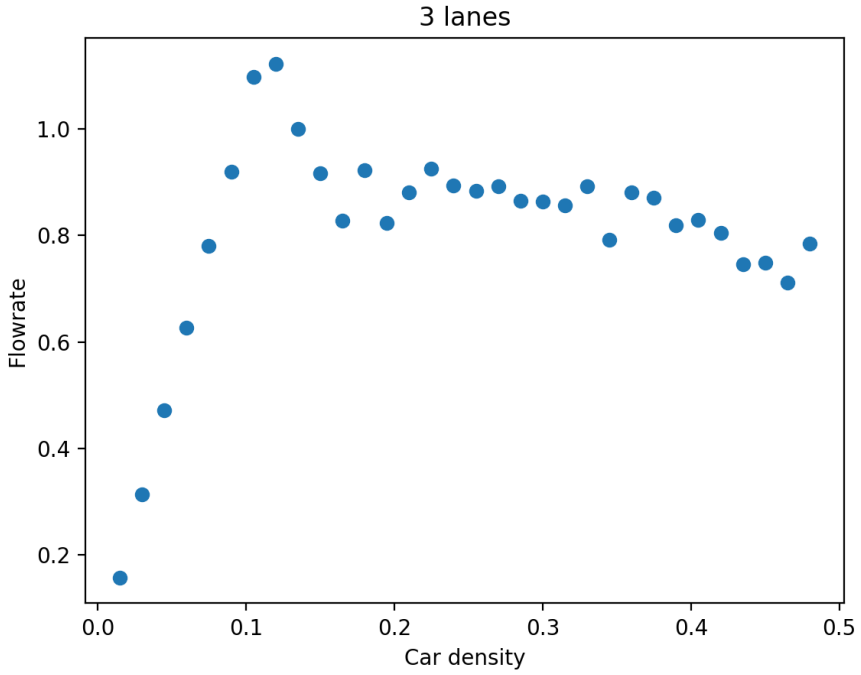
I ran the simulation above for 300 timesteps for each car density (represented on the x axis). I measured the flowrate at every timestep from 240 to 299, what you see on the y axis in the plot above is the average of those points flowrate. The maximum flowrate above is when the car density is 0,12 (12

cars). Below is the statistical error measured over the time steps from 240 to 299



As one can see, the error is pretty low and we can with high probability say that a car density of 0,12 (standard error 0,007-0,008) gives the best flowrate.

Comparison 3 lanes, roadlength 100 and roadlength 200:



Above is for road length 200 and just like before the flowrate is the biggest at car density 0,12. You could have taken any road length and it would have given the same result.

Conclusion

The results I have obtained are very reasonable and interesting. It makes sense that more lanes would lead to a maximum flowrate at a bigger car density. In a three lane highway that is 278m where every time step is 1 second and cars are going between 100-120 km/h, 12 cars gives the best traffic flow. If there would be 4 cars in each lane, meaning the distance between each car (if they are spread out evenly) is $\frac{real\ road\ length}{4} = 69m$, cars have on average a maximum velocity of 110km/h meaning there is about 2,3s between cars on average. This is very reasonable and it could very well represent best flow on a three lane highway. When you start adding more cars at a stationary point, it could give a positive contribution to the flowrate as long as it does not affect the other cars velocities to much. That is why the plots above look the way they do, after a certain car density the flowrate starts to decrease because that addition affects the other cars velocities and that decreases the flowrate.

The interesting part is that the maximum flowrate, which for 3 lanes is at 12 cars, is lower than if we had 3 separate 1 lane highways where every car had $v_{max} = 10$ and no lane changes were allowed. If you look at the graph for the 1 lane highway you can see that the maximum flowrate for 6 cars clearly is higher than 0.55. If you look at the graph for the 3 lane highway you can see that the maximum flowrate, which is for 12 cars, is about 1,2. This means that if we add 6 cars with $v_{max} = 10$ in each lane in a 3 lane highway where lane-changing is not allowed. That flowrate would be better than the flowrate of the three lane highway where lane-changing is allowed:
$3 \cdot 0,55 = 1,65 > 1,2$ and it would be at a higher car density of $3 \cdot 6/100 = 18/100 > 12/100$. This means that drivers on the 3 lane highway where lane changing is allowed are actually messing up for cars behind when they are switching to the lane that has the longest distance to the next car. The driver does not consider cars other than the next car with the longest distance in front. I think the model I have set up gives an okay model for real traffic flow. Big trucks that are going in a highway are not allowed to go over a certain speed limit that is lower than that of the highway, this could be represented by cars with lower values on v_{max} in my model. But sometimes

those drivers still choose to drive around other trucks because the driver wants to get forward as fast as possible and does not take into account the overall traffic flow. In a highway where drivers had that mindset I would say that this model is accurate but that is usually not the case. It lacks in that it does not take into account that cars in the left lane generally goes faster than cars in the right lane

Python code simulation for 12 cars going in 3 lanes on a highway with road length 100:

```
from matplotlib import pyplot as plt
import random
from matplotlib import animation
from operator import attrgetter
import pdb

roadLength    = 100
numFrames     = 2000
numCars = 12
probability = 0.5



class Car:
    def __init__(self, position, lane, vmax):
        self.position = position
        self.velocity = 0
        self.vmax = vmax
        self.lane = lane
        self.nextlane1 = None
        self.nextlane2 = None
        self.nextlane3 = None
```

```
block11 = []
block12 = []
block21 = []
block22 = []
block31 = []
block32 = []
block41 = []
block42 = []
block51 = []
block52 = []
block61 = []
block62 = []

def pos(car,k):
    rand = random.uniform(0,1)
    if car[k].velocity < car[k].vmax:
        car[k].velocity +=  1
    struts = 0
    while True:
        struts += 1

        if car[k].lane == 1 and car[k].nextlane1 and car[k].nextlane2:
            block11.append(1)
            temp = [car[k].nextlane1, car[k].nextlane2]
            nexlist = [car[k].nextlane1.position - car[k].position,
car[k].nextlane2.position - car[k].position]
            if nexlist[0] < 0:
                nexlist[0] = roadLength + nexlist[0]
            if nexlist[1] < 0:
                nexlist[1] = roadLength + nexlist[1]
            nextdist = max(nexlist)
            nextlane = []
            for p in range(len(nexlist)):
                if nexlist[p] == nextdist:
                    nextlane.append(p)

            if nextdist <= car[k].velocity:
                car[k].velocity = nextdist - 1
            car[k].lane = random.choice(nextlane) + 1
            block12.append(car[k].lane)
            break

        if car[k].lane == 1 and (car[k].nextlane1==None or
car[k].nextlane2==None):
            block21.append(1)
            temp = []
            if car[k].nextlane1==None:
                temp.append(1)
            if car[k].nextlane2==None:
                temp.append(2)
            car[k].lane = random.choice(temp)
            block22.append(car[k].lane)
            break
```

```
        if car[k].lane == 2 and car[k].nextlane1 and car[k].nextlane2 and
car[k].nextlane3:
            block31.append(1)
            temp = [car[k].nextlane1, car[k].nextlane2, car[k].nextlane3]
            nexlist = [car[k].nextlane1.position - car[k].position,
car[k].nextlane2.position - car[k].position, car[k].nextlane3.position -
car[k].position]
            if nexlist[0] < 0:
                nexlist[0] = roadLength + nexlist[0]
            if nexlist[1] < 0:
                nexlist[1] = roadLength + nexlist[1]
            if nexlist[2] < 0:
                nexlist[2] = roadLength + nexlist[2]
            nextdist = max(nexlist)
            nextlane=[]
            for p in range(len(nexlist)):
                if nexlist[p] == nextdist:
                    nextlane.append(p)

            if nextdist <= car[k].velocity:
                car[k].velocity = nextdist - 1
            car[k].lane = random.choice(nextlane) + 1
            block32.append(car[k].lane)
            break


        if car[k].lane == 2 and (car[k].nextlane1==None or
car[k].nextlane2==None or car[k].nextlane3==None):
            block41.append(1)
            temp = []
            if car[k].nextlane1==None:
                temp.append(1)
            if car[k].nextlane2==None:
                temp.append(2)
            if car[k].nextlane3==None:
                temp.append(3)
            car[k].lane = random.choice(temp)
            block42.append(car[k].lane)
            break


        if car[k].lane == 3 and car[k].nextlane2 and car[k].nextlane3:
            block51.append(1)
            temp = [car[k].nextlane2, car[k].nextlane3]
            nexlist = [car[k].nextlane2.position - car[k].position,
car[k].nextlane3.position - car[k].position]
            if nexlist[0] < 0:
                nexlist[0] = roadLength + nexlist[0]
            if nexlist[1] < 0:
                nexlist[1] = roadLength + nexlist[1]
            nextdist = max(nexlist)
            nextlane = []
            for p in range(len(nexlist)):
                if nexlist[p] == nextdist:
```

```
                    nextlane.append(p)

            if nextdist <= car[k].velocity:
                car[k].velocity = nextdist - 1
            car[k].lane = random.choice(nextlane) + 2
            block52.append(car[k].lane)
            break

        if car[k].lane == 3 and (car[k].nextlane2==None or
car[k].nextlane3==None):
            block61.append(1)
            temp = []
            if car[k].nextlane2==None:
                temp.append(2)
            if car[k].nextlane3==None:
                temp.append(3)
            car[k].lane = random.choice(temp)
            block62.append(car[k].lane)
            break

        else:
            pdb.set_trace()

    if rand < probability and car[k].velocity > 0:
        car[k].velocity = car[k].velocity - 1
    car[k].position = (car[k].position + car[k].velocity) % roadLength
    retur = f(cars)
    return retur


# Point the attribute nextlane1, nextlane2 and nextlane3 to the next cars in
each of those lanes
def f(carr):
    sort_cars = sorted(carr, key=attrgetter("position"))
    iterator1=[]
    for l in range(numCars):
        iterator1.append(l)

    iterator2=[]
    for s in range(numCars):
        iterator2=[]
        for p in range(s+1, numCars):
            iterator2.append(p)
        for t in range(s):
            iterator2.append(t)
        iterator1[s] = iterator2.copy()


    for p in range(numCars):
        countl1 = 0
        countl2 = 0
        countl3 = 0

        if sort_cars[p].position == sort_cars[p-1].position:
            if sort_cars[p-1].lane == 1:
```

```
                sort_cars[p].nextlane1 = sort_cars[p-1]
                countl1 = 1
            if sort_cars[p-1].lane == 2:
                sort_cars[p].nextlane2 = sort_cars[p-1]
                countl2 = 1
            if sort_cars[p-1].lane == 3:
                sort_cars[p].nextlane3 = sort_cars[p-1]
                countl3 = 1
            if sort_cars[p].position == sort_cars[p-2].position:
                if sort_cars[p-1].lane == 1:
                    sort_cars[p].nextlane1 = sort_cars[p-2]
                    countl1 = 1
                if sort_cars[p-1].lane == 2:
                    sort_cars[p].nextlane2 = sort_cars[p-2]
                    countl2 = 1
                if sort_cars[p-1].lane == 3:
                    sort_cars[p].nextlane3 = sort_cars[p-2]
                    countl3 = 1
        for j in iterator1[p]:
            if sort_cars[j].lane ==1 and countl1 == 0:
                sort_cars[p].nextlane1 = sort_cars[j]
                countl1 = 1
            if sort_cars[j].lane ==2 and countl2 == 0:
                sort_cars[p].nextlane2 = sort_cars[j]
                countl2 = 1
            if sort_cars[j].lane ==3 and countl3 == 0:
                sort_cars[p].nextlane3 = sort_cars[j]
                countl3 = 1
            if countl1 == 1 and countl2 == 1 and countl3 == 1:
                break
        if countl1 == 0:
           sort_cars[p].nextlane1 = None
        if countl2 == 0:
           sort_cars[p].nextlane2 = None
        if countl3 == 0:
           sort_cars[p].nextlane3 = None
    return sort_cars



def animate(frameNr):
    position = []
    yaxlane=[]
    sort_cars = sorted(cars, key=attrgetter("position"))
    for i in range(numCars):
        sort_cars = pos(sort_cars,i)
    for k in range(numCars):
        position.append(cars[k].position)
        yaxlane.append(cars[k].lane)
    point.set_data(position, yaxlane)
    return point,
```

```
start=[]
cars =[]
pos1 = 0
pos2 = 0
pos3 = 0
lc1 = 0
lc2= 1
lc3 = 3
for st in range(numCars):
    if st % 3 == 0:
        cars.append(Car(pos1, lc1 % 3+1, 10))
        pos1 += 1
        lc1 += 1

    if st % 3 == 1:
        cars.append(Car(pos2, lc2 % 3+1, 11))
        pos2 += 1
        lc2 += 1

    if st % 3 == 2:
        cars.append(Car(pos3, lc3 % 3 +1, 12))
        pos3 += 1
        lc3 += 1
cars = f(cars)



#
fig = plt.figure()
ax = fig.add_subplot(111)
point, = ax.plot(start, start, 'o')
ax.set_xlim(-5, 200)
ax.set_ylim(0, 4)

def simulate():
    anim = animation.FuncAnimation(fig, animate,
                                   frames=numFrames, interval=1000, blit=True,
repeat=False)

    plt.show()

simulate()




print('Block12, 1 = ' + str(block12.count(1)) + ' 2 = ' +
str(block12.count(2)) + ' 3 = ' + str(block12.count(3)))
print('Block22, 1 = ' + str(block22.count(1)) + ' 2 = ' +
str(block22.count(2)) + ' 3 = ' + str(block22.count(3)))
print('Block32, 1 = ' + str(block32.count(1)) + ' 2 = ' +
str(block32.count(2)) + ' 3 = ' + str(block32.count(3)))
print('Block42, 1 = ' + str(block42.count(1)) + ' 2 = ' +
str(block42.count(2)) + ' 3 = ' + str(block42.count(3)))
print('Block52, 1 = ' + str(block52.count(1)) + ' 2 = ' +
```

```
str(block52.count(2)) + ' 3 = ' + str(block52.count(3)))
print('Block62, 1 = ' + str(block62.count(1)) + ' 2 = ' +
str(block62.count(2)) + ' 3 = ' + str(block62.count(3)))

print('Block11, 1 = ' + str(block11.count(1)))
print('Block21, 1 = ' + str(block21.count(1)))
print('Block31, 1 = ' + str(block31.count(1)))
print('Block41, 1 = ' + str(block41.count(1)))
print('Block51, 1 = ' + str(block51.count(1)))
print('Block61, 1 = ' + str(block61.count(1)))
```