Modelling, Simulation and Optimization of Multi-modal Motorway Traffic Flow

Rohan Naryan Koli
MSc Data Analytics
National College of Ireland
x19224842@student.ncirl.ie

***Abstract*—Traffic congestion is one of the most serios problems in urban cities. This research paper simulates the movement of traffic in Urban environment and also provides lane merging simulation of multi-modal traffic.**

***Index Terms*—Simulation, SimPy, Traffic, Autonomous cars, Congestion**

I. Introduction

Roadways are an important form of a country's infrastructure which provide movement of people and goods from a place to another. They provide access to employment, health and education services, essential goods, social visits which ultimately leads to socio-economical development. Hence, it is critical to maintain the laminar flow of roadways in an uninterrupted manner. Given the rise in population, purchasing power and technological advances there has been a rise in traffic congestion levels across the globe. Like a river, traffic congestion is one of the most complex problems faced by metropolitan cities which further bifurcates into different problem streams such as pollution, stress levels, costs (business costs, fuel costs and economical loss), poor quality of life, inward investment.

In 2020, among the most congested cities, Dublin was ranked 17th most congested city [1] in the world. Secondly, in 2018, the cost of congestion in Ireland was 1.1% of GDP which totals massively to €1.8 billion[2]. Given the deteriorating road conditions and increased congestion, traffic management has become the need of the hour.

In this research, we model and simulate motorways two-lane motorways. Secondly, we simulate a converging road scenario in which the motorway converges from three lanes to two lanes. While simulating the motorways, we investigate and optimize the results by analysing the through puts, average travel times and minimize the scenario in which "crash" events have been occurred. The modelling, simulation and optimization is also tested on various vehicles and driver behaviours to match real-time scenarios.

II. Methodology

The simulations are generated using python as the base coding language using Google colab. For simulation we have used the python library "SimPy" which has a discrete-event simulation framework to simulate results in real-time. Google Colab is provides a disk space of 200 gb and a RAM of 25 gb.

Broadly, the initial modelling consists of assigning critical entities and creating three classes namely, for lanes, for vehicle and for creating simulation recordings. The critical entities defines variables such as lane changing times, critical response times, minimum speed difference for overtaking the car ahead, length of the car and lastly the safe distances ahead and behind for triggering the overtake events.

The *lanes* class creates initial lanes for simulation. Along with lane creation, the lanes class provides features to widen, extend lanes both in the left as well as on the right of the current lane. Lastly, sub-classes "enter" and "leave" records the events when a vehicle enters or leaves a particular lane respectively. The first lane on the right is marked as the fast lane or the overtaking lane, where as the middle lane is programmed as driving lane.

The class *vehicle* defines sub-classes update, update-only, the process, emergency braking, change lanes and adjust velocity. These classes defines the behavior of vehicles except the classes update and update only, which updates the vehicle attributes at the point of time to the recorder. Secondly, the sub-class surround acts as an awareness system for the vehicle as shown in fig.1.

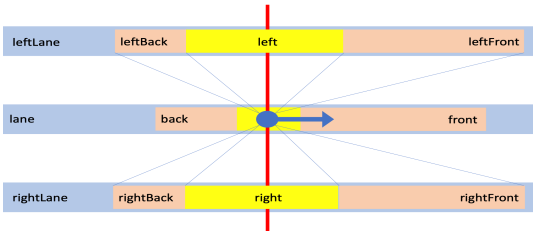

Fig. 1. Vehicle Surroundings

Lastly, the *recorder* class records the simulation of all the cars in the lane, along with their behaviors using the SimPy environment. We have defined the sub-classes to dataframe of recorded variables which integrates all the actions of classes vehicle, lanes and critical entities. Apart from recording, the recorder class also provides plotting functionality to visualize the simulation results.

[1]https://www.tomtom.com/en_gb/traffic-index/dublin-traffic/#statistics
[2]https://www.irishexaminer.com/news/arid-30970679.html

III. The Simulation Model

In the below sections we simulate the results for different scenarios and optimize the results to address the traffic congestion problem with pragmatic solutions. The initial parameters are defined as in fig.2: We select the critical tolerance time as

```
CRITICAL_TIME_TOLERANCE = 4  # [s]
LANE_CHANGE_TIME = 2 # [s]
MIN_TIME_DIFF = 0.5

MIN_SPEED_DIFF = 3 # [m/s] min speed diff to trigger overtaking

CAR_LENGTH = 4 # [m]

FAR_AWAY_IN_FRONT = 200 # [m]  distance at which a car in front can be ignored
FAR_AWAY_IN_BACK = 80   # [m]  distance at which a car behind can be ignored
```

Fig. 2. Initial parameters

4 seconds which is universally accepted as the stoppage time for worst driving conditions. Next, we define lane changing time, minimum time difference, minimum speed difference for overtaking as 2 , 0.5 and 3 seconds respectively. We have selected the car length as 4 metre (average length of a sedan). Lastly, we have selected the ignorance distance at the front and back as 200 and 80 metre respectively.

A. Initial Lane Modelling and Testing

1) Two lanes and Fixed Speed:

In the first part, we test the motorway with two 3 kilometer lanes as being non-congested to determine simulation results. We use the lanes class to create two lanes as depicted in fig.3. The Road Safety Authority of Ireland (RSA) suggests five

Multiple Vehicles with Fixed Speed

```
#Creating Lanes
l = Lane(3000, VMAX)
r = l.widenRight()
print("Left Lane: ", l)
print("Right Lane:", r)
```

```
Left Lane:  [4 3000m R:5]
Right Lane: [5 3000m L:4]
```

Fig. 3. Lane Creation

speed limits depending upon the road types as shown in the table III-A1[3].

Road Classification	Speed
Motorways	120km/h
National roads	100km/h
Regional roads	80km/h
Urban areas	50km/h
Special limits (schools etc)	30km/h

TABLE I
Road Classification

We created a list of five different speed limits to calculate the through-puts and average travel times for each maximum speed limits. Since hypothetically, we have considered a non-congested lane, the vehicle speed is considered to be constant and equal to the maximum speed limits(lane speed = VMAX) with constant acceleration. We defined number of cars 200 and inter-arrival time (DT) as per *4-second rule* which is defined by the RSA as a safe headway to ensuring we are at least two seconds behind the vehicle in front to display safe braking without harm.

```
for x in a:
  VMAX = x
  N = 100
  DT = 2 # time difference between start
  env = simpy.Environment()
  rec = SimpleRecorder(env, 0, 300, 1)

  l = Lane(3000, VMAX)
  r = l.widenRight()
  for i in range(N):
    v = Vehicle(env, rec, startingLane=l, t0=i*DT,dx0=VMAX)
    v = Vehicle(env, rec, startingLane=r, t0=i*DT,dx0=VMAX)
    #v.traceOvertake = True
  rec.run()
```

Fig. 4. Defining Vehicle and Environment

For calculating the average time, we stored the recorder result in a dataframe, and extracted the key value pairs of each vehicle to get their start and end times using the ”id” column storing them in ”start” and ”end”. Next, we get a difference of each vehicle entering and leaving lane and divide by the length of the dataframe(which is also equal to the number of vehicles entering or leaving the lane).

```
#Average time calculation
df = rec.getData()
start, end = {},{}
time_taken = []
for i in range(len(df)):
  if df["id"][i] not in start:
    start[df["id"][i]] = df["t"][i]
    end[df["id"][i]] = df["t"][i]
  end[df["id"][i]] = df["t"][i]

for i in start:
  time_taken.append(end[i]-start[i])
  #print(start)
  #print(end)
average = sum(time_taken) / len(time_taken)
average_time.append(average)
print(f"The Average time for speed limit {VMAX} kmph is {average} seconds")
```

Fig. 5. Average time calculation

Throughput is an indicator of the productivity of the system. It tells the us how many vehicles were processed by the system during the analysis period. The US department of transportation defines throughput as - ”the number of distinct vehicles (or people) able to enter or exit the system during the analysis period.[4]”

```
#Throughputs calculation
event = "end"
for index, row in df.iloc[::-1].iterrows():
  if df["event"][index]=="end":
    number = df["id"][index]
    time = df["t"][index]
    break
factor = 3600/time
throughput = number*factor
throughput_saves.append(throughput)
print(f"and the Throughput is {throughput} cars per hour")
```

Fig. 6. Throughput calculation

In our simulation, we calculate the through-put by recording the event of first car entering the lane and the last car coming out of the lane and normalizing the results to an hours time.

The results along with the average times and throughput is depicted in fig.7.

[3]https://www.rsa.ie/en/RSA/Licensed-Drivers/Driving-in-Ireland/

[4]https://ops.fhwa.dot.gov/publications/fhwahop08054/sect6.htm

	Speed_Limits	Average_Time	Throughput
0	13.888889	216.0	216.0
1	22.222222	136.0	136.0
2	27.777778	108.0	108.0
3	33.333333	90.9	90.9

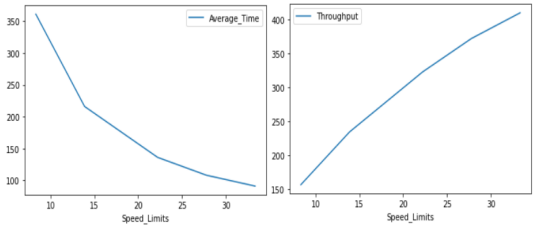

Fig. 7. Non-congested lane results

Basic observation suggests that, with the increase in speed limits of the motorway, the average travel times are reduced and the throughput of vehicles (per hour) is increased.

2) Two lanes and Variable Speed with time:

In this section, we define random variation in speed along with random time for changing the speed. We use the function *random.normalvariate* from the python library random to generate a normal distribution using mean and standard distribution. Next, we define two functions to generate random speeds in the same number of random intervals as shown in fig.8.

```
[56] SLOW_CYCLE = 200

[105] def randomIntervals(cycles):
        # return [ random.expovariate(1.0/SLOW_CYCLE)+10 for i in range(cycles) ]
        return [ max(0, random.normalvariate(SLOW_CYCLE, SLOW_CYCLE/3)) for i in range(cycles) ]

SPEED_VARIATION = 0.10
def randomSpeedVariation(vmax, cycles, cv=SPEED_VARIATION):
    return [ vmax + (-1)**i*abs(random.normalvariate(0, vmax*cv)) for i in range(cycles) ]
```

Fig. 8. Speed and Interval Variation

lastly, to simulate the results for we use a speed variation factor of 0.05 with average vehicle speed of 120 kmph (which is randomly assigned using normal distribution function as suggested above) and changing random time intervals between 1 and 3 and simulating for 80 cars as shown in the code in fig.9. The cars were travelling on the primary slow lane, and only overtaking the car ahead using the fast lane and finally maneuvering back to the slow lane. The simulation results show an average travel time of 118 seconds and a throughput of 569 cars per hour.

Fig,10 depicts the graphs of all 80 vehicles simulated and their behaviors such as overtaking and changing speed profiles. The black arrows pointing right indicate, vehicles changing lanes towards the fast lane. The red circles indicate the braking

```
VMAX = 120/3.6
N = 80 # number of points
DT =  4# time difference between start
random.seed(13)
env = simpy.Environment()
rec = simpleRecorder(env, 0, 1000, 1)

for i in range(N):
    CYCLES = random.randint(1, 3)
    times = randomIntervals(CYCLES)
    #speed = randomSpeedVariation(VMAX+i, CYCLES)
    speed = randomSpeedVariation(VMAX, CYCLES)
    v = Vehicle(env, rec, startingLane=1, t0=i*DT, dx0=speed[-1], t=times, v=speed)
    #v = Vehicle(env, rec, startingLane=r, t0=i*DT, dx0=speed[-1], t=times, v=speed)
    v.traceOvertake = True
    #v.traceSurround = True
rec.run()
```

```
t=  115.0s Overtaking v21 overtakes v20 at x=  805.4m
t=  160.0s Overtaking v24 overtakes v23 at x=1,747.7m
t=  160.0s Overtaking v29 overtakes v28 at x=1,080.9m
t=  181.0s Overtaking v25 overtakes v23 at x=2,263.4m
t=  201.0s Overtaking v25 returns to slow lane at x=2,930.7m
t=  221.0s Overtaking v31 overtakes v30 at x=2,542.2m
t=  226.0s Overtaking v52 overtakes v51 at x=  293.0m
t=  232.0s Overtaking v31 returns to slow lane at x=2,905.7m
t=  241.0s Overtaking v34 overtakes v33 at x=2,550.9m
t=  247.0s Overtaking v45 overtakes v44 at x=1,251.9m
t=  253.0s Overtaking v34 returns to slow lane at x=2,943.8m
t=  290.0s Overtaking v58 overtakes v57 at x=  854.5m
t=  307.0s Overtaking v74 overtakes v73 at x=  152.7m
```

Fig. 9. Simulation with Random Speed, Time interval

mechanism. The black diamond marking indicate the vehicle is out of the lane. Lastly, the cross mark indicates a crash event has occured.

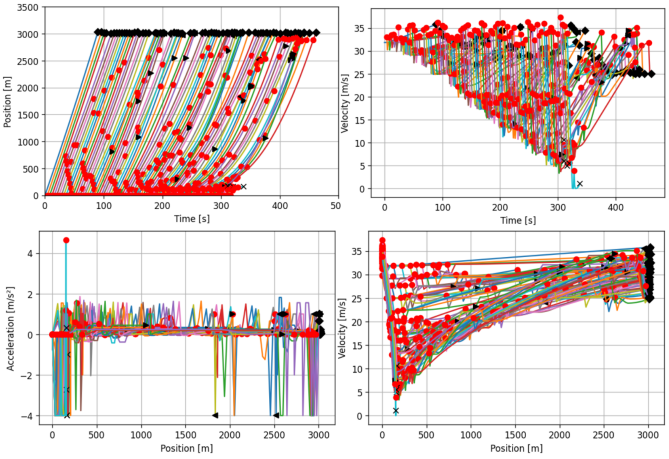

Fig. 10. Simulation graphs for Random Speed, Time interval

B. Simulation based on Different Car models

In this section we have created three types of vehicles to simulate different results. The first class of vehicles are the passenger vehicles which were used in the previous simulations with default values. The second class of vehicles include commercial vehicles, which have a greater length (10m) compared to the passenger vehicles. Lastly, we have included sports vehicles to simulate fast cars with greater accelerations/deceleration's and velocity. The details of the parameters are given in the table below-

1) Based on Fixed Speed: On running the simulation for different vehicles with fixed time and speed intervals, we get an average travel time of 142 seconds whereas a throughput of 210 seconds. On examining the graphs in fig.11, we find many vehicles changing lanes. This may be due to the inclusion of sports cars which will overtake the slow cars as per the attributes. Also, inclusion of heavy vehicles causes many passenger and sports vehicles to change lanes and heavy braking on the motorways.

Vehicle Type	Passenger	Commercial	Sports
CAR_LENGTH (m)	4.5	10	1.5
LANE_CHANGE_TIME (sec)	2	5	5
FAR_AWAY_IN_FRONT (sec)	200	100	400
FAR_AWAY_IN_BACK (sec)	80	40	160
Speed (m/s)	20	14	30
Acceleration (m/s2)	3	0.75	7.5
Deceleration(m/s2)	-4.3	-0.85	-8

TABLE II
VEHICLE PARAMETERS

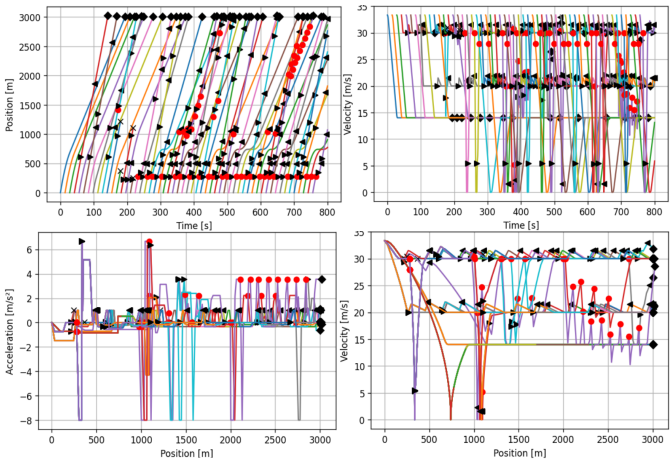

Fig. 11. Simulation: Fixed Speed with Random Vehicle

2) Based on Variable Speed, Time interval: To analyze the different vehicles with varying speed and time intervals, we combine the code structure as below as in fig.12.

```
for i in range(N):
    CYCLES = random.randint(1, 3)
    times = randomIntervals(CYCLES)
    speed = randomSpeedVariation(VMAX, CYCLES)
    n = random.randint(1,3)
    if n==1:
      CAR_LENGTH = 4.5
      LANE_CHANGE_TIME = 2
      FAR_AWAY_IN_FRONT = 200 # [m]  distance at which a car in front can be ignored
      FAR_AWAY_IN_BACK = 80   # [m]  distance at which a car behind can be ignored
      v = Vehicle(env, rec, "passenger",startingLane=1, t0=i*DT, dx0=speed[-1], t=min(times,[10]), v=min(speed,[20]))
    elif n==2:
      CAR_LENGTH = 10
      LANE_CHANGE_TIME = 5
      FAR_AWAY_IN_FRONT = 100 # [m]  distance at which a car in front can be ignored
      FAR_AWAY_IN_BACK = 40   # [m]  distance at which a car behind can be ignored
      v = Vehicle(env, rec, "commercialV",startingLane=1, t0=i*DT, dx0=VMAX, t=min(times,[30]), v=min(speed,[14]))
    else:
      LANE_CHANGE_TIME = 1.5
      CAR_LENGTH = 5
      FAR_AWAY_IN_FRONT = 400 # [m]  distance at which a car in front can be ignored
      FAR_AWAY_IN_BACK = 160   # [m]  distance at which a car behind can be ignored
      v = Vehicle(env, rec, "sports",startingLane=1, t0=i*DT, dx0=VMAX, t=min(times,[5]), v=min(speed,[30]))

    v.traceOvertake = True
```

Fig. 12. Code: Mixed Vehicles, Random Speed and Time interval

On examining, we get an average travel time of 152 seconds, where as the though-put has been decreased to 190 cars per hour. Variation in speed captures the erratic behavior of drivers and hence the decrease in throughput is inevitable.

The inter-arrival time (DT) has been increased to 15 seconds from 14 seconds as the simulation results included many crash events.

C. Simulating Motorway Convergence

To simulate the Motorway convergence, we define a function in the *vehicle class* wherein we check if the lane is available ahead of the vehicle at a distance of 500m. Next we

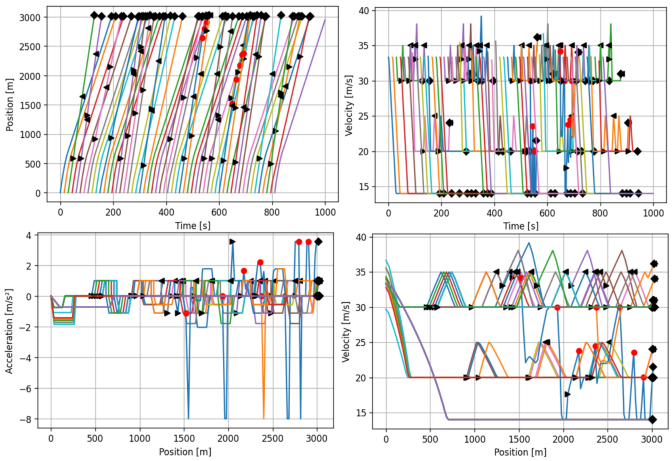

Fig. 13. Simulation Graph: Different Vehicles, Random Speed and time interval

check if the rightlane, leftlane is available to overtake. Also we check if the lenghth of the right lane is greater than 500 m to avoid crash.Lastly, we add the self.MergeToRight function in the explicit change of lane condition.

Below is the function to merge to the right lane

```
def MergeToRight(self):
    if self.lane.length-self.pos <= 500 and \
        self.surround.rightLane is not None and \
        self.surround.leftLane  is None and \
        self.surround.rightLane.length >=500:
            return True
    else:
        return False
```

Fig. 14. Merge Lane code

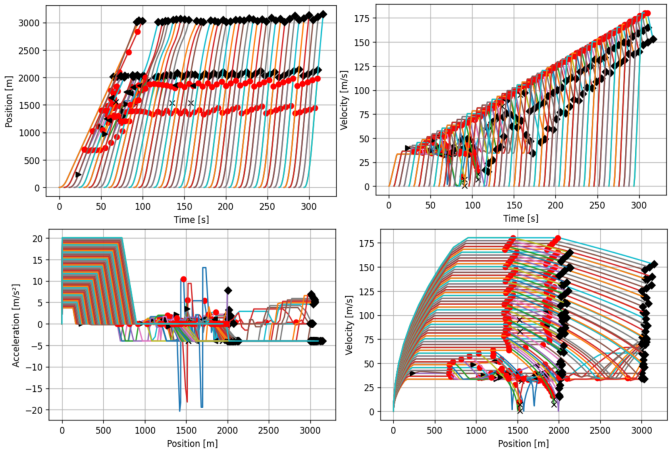

Fig. 15. Merge Lane Graphs

The simulation results suggest the average travel time as 35 secs and throughput of 1124 cars per hour. Fig.15 provides the graphical representation of the simulation with fixed speed.

IV. Conclusion

We were able to provide a traffic multi-modal simulation for the cities. Such systems will help us in improving the traffic conditions of cities and ultimately help in economic as well as social development with enhanced connectivity and by minimizing business losses.