Notes:

General notes for how reward is calculated.

There are 3 buffers, a NS buffer, and EW buffer and a free buffer. NS stores all the cars on the road running North-South, EW stores all the cars for East-West, and free buffer stores the cars that have passed through the intersection. When the light is green, we calculate the reward by multiplying each buffer's average wait time by the number of cars. The total reward is the difference between the two, with the positive/negative being relative to the phase of the light, additionally, the amount of cars allowed to pass through impacts the reward positivly.

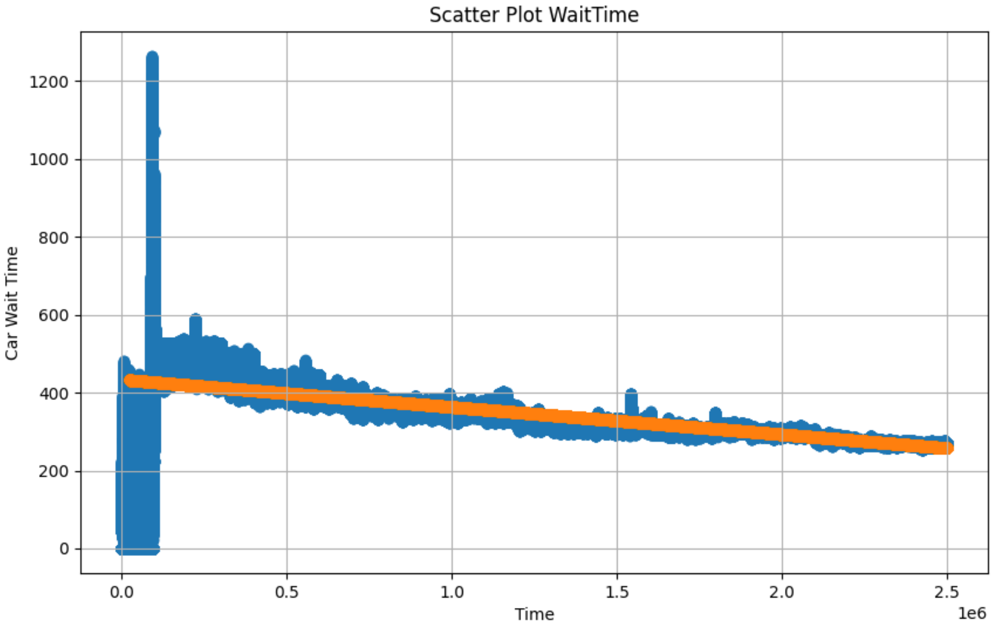
Reward calculation is done by

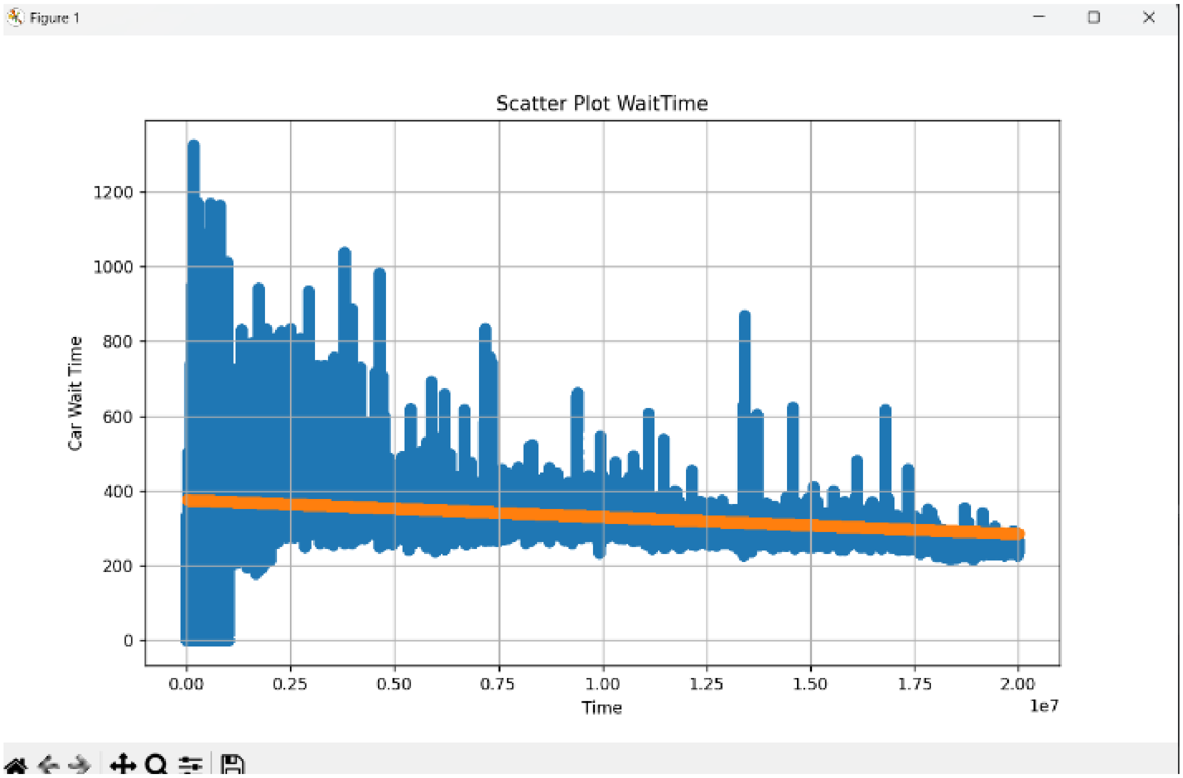
```
if cur_action == LightAction.V_GREEN:
        tot = (self.pos_scale * (ns_cars * ns_wait) - self.neg_scale*(ew_cars * ew_wait))
        tot += self.buffer_scale * self.buffer_reward(buffer)
      elif cur_action == LightAction.H_GREEN:
        tot = (self.pos_scale * (ew_cars * ew_wait) - self.neg_scale*(ns_cars * ns_wait))
        tot += self.buffer_scale * self.buffer_reward(buffer)

      elif cur_action == LightAction.H_YELLOW or cur_action == LightAction.V_YELLOW:
        tot = 0.001 * self.neg_scale * (-(ew_cars * ew_wait) - (ns_wait * ns_cars))
      else:
        tot = 0.01 * self.neg_scale * (-(ew_cars * ew_wait) - (ns_wait * ns_cars))

      return tot, (ns_wait + ew_wait)/2
```

Unless stated otherwise, gamma=0.9,step_size=0.9,epsilon=0.01,discount_rate=0.99

Experiment 1:

pos_scale = 2
neg_scale = 1.5
buffer_scale = 3



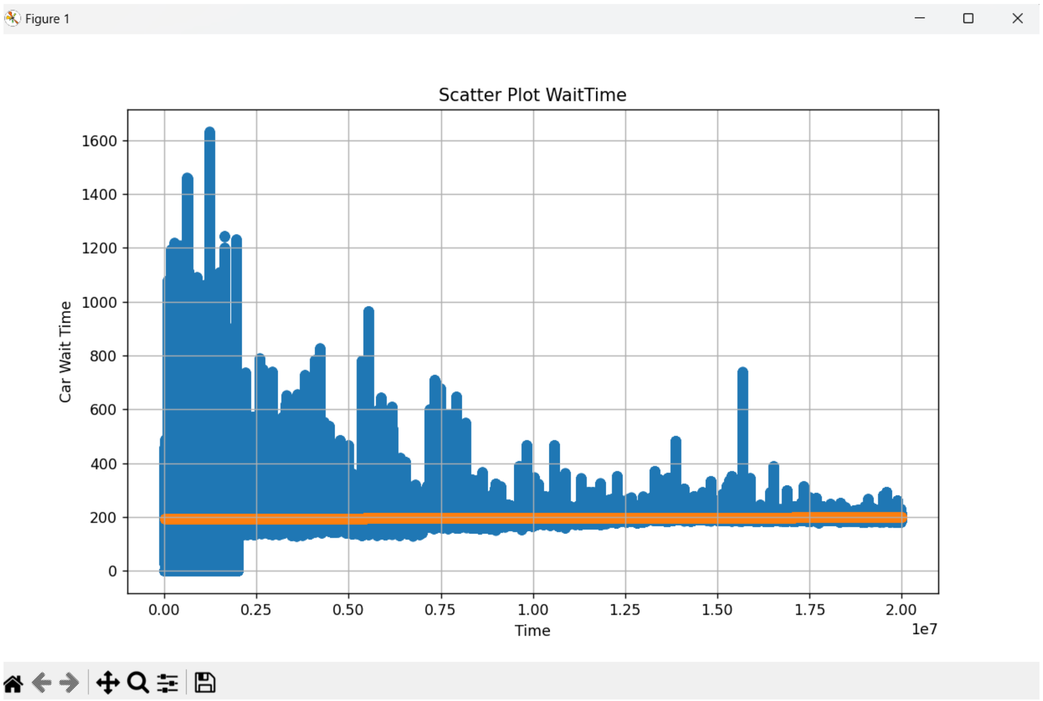


Experiment 2:

pos_scale = 2

neg_scale = 1.5

buffer_scale = 3

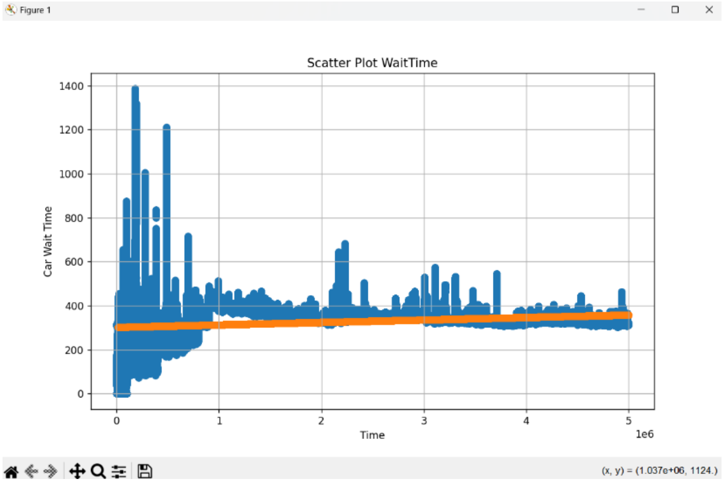


Experiment 3ish:

pos_scale = 1.5

neg_scale = 2

buffer_scale = 3

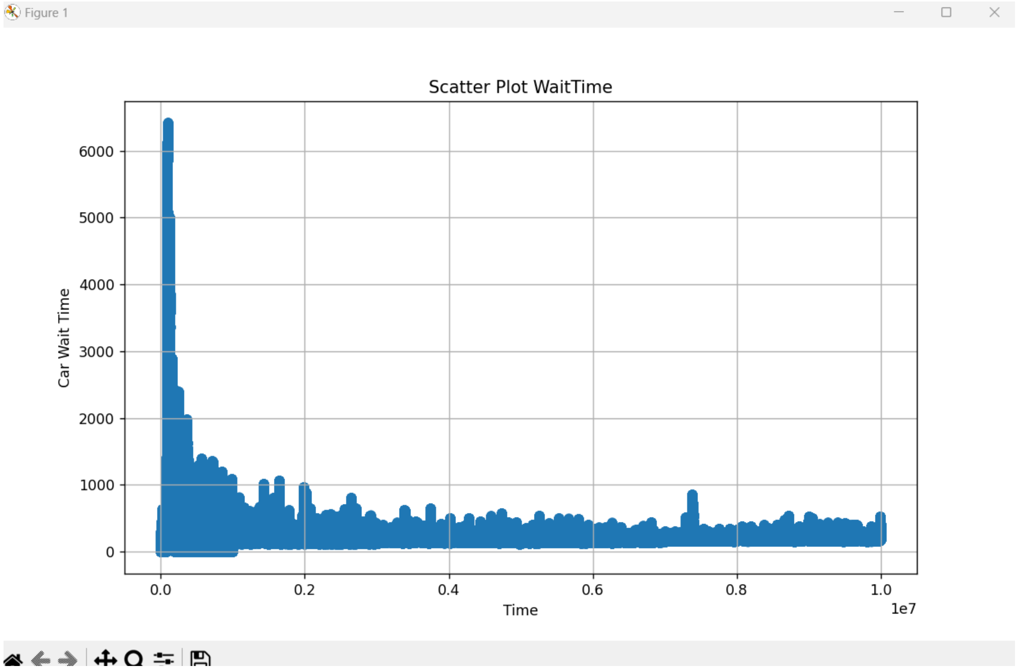


(x, y) = (1.037e+06, 1124.)

Experiment 4:

Epsilon = 0.01
Step_size =0.7

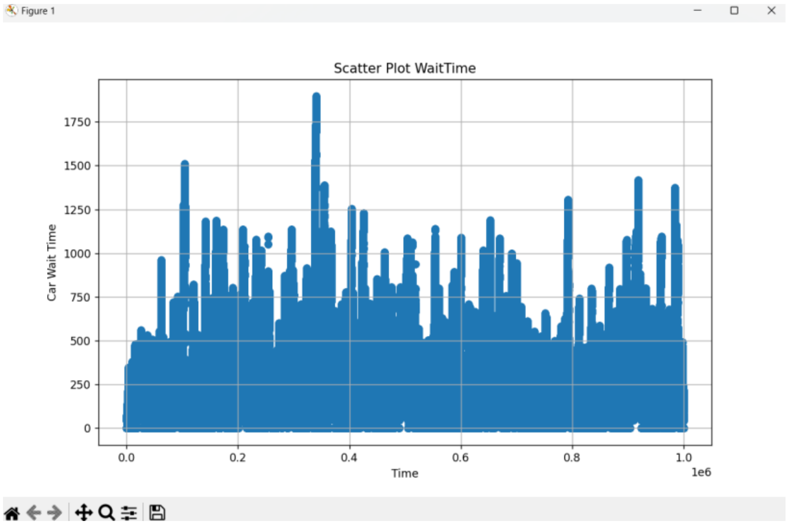
pos_scale = 1
neg_scale = 0.5
buffer_scale = 3



Experiment 5:

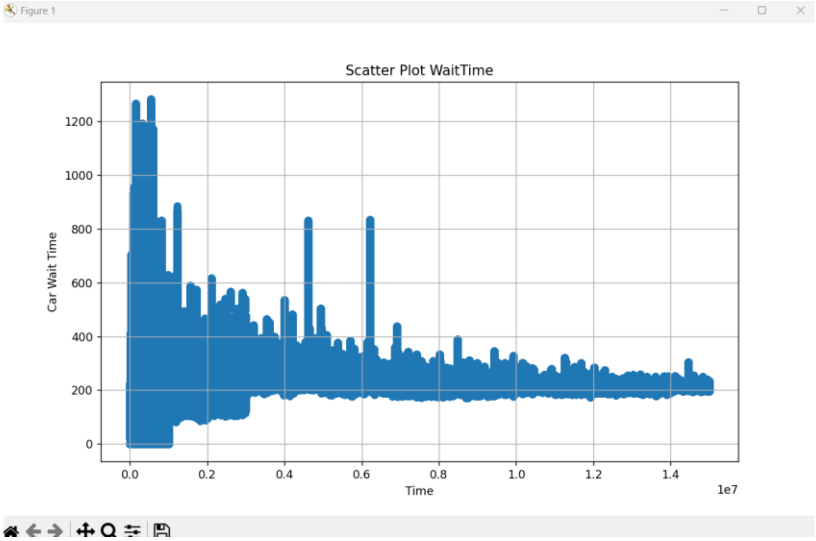
step_size =0.01

pos_scale = 1
neg_scale = 0.5
buffer_scale = 3



Experment 6:

pos_scale = 1
neg_scale = 0.5
buffer_scale = 9



New Reward Alg:

```
if cur_action == LightAction.V_GREEN:
   tot = (self.pos_scale * (ns_wait) - self.neg_scale * (ew_wait))
   tot += self.buffer_scale * self.buffer_reward(buffer)
elif cur_action == LightAction.H_GREEN:
   tot = (self.pos_scale * (ew_wait) - self.neg_scale * (ns_wait))
   tot += self.buffer_scale * self.buffer_reward(buffer)

# elif cur_action == LightAction.H_YELLOW or cur_action == LightAction.V_YELLOW:
#     tot = (1-act) * self.neg_scale * (-(ew_cars * ew_wait) - (ns_wait * ns_cars))
else:
   tot = (1-act) * self.neg_scale * (-(ew_wait) - (ns_cars))

return tot, (ns_wait + ew_wait) / 2
```

Experiment 1:

Step_size = 0.9

pos_scale = 1.5
neg_scale = 2
buffer_scale = 3

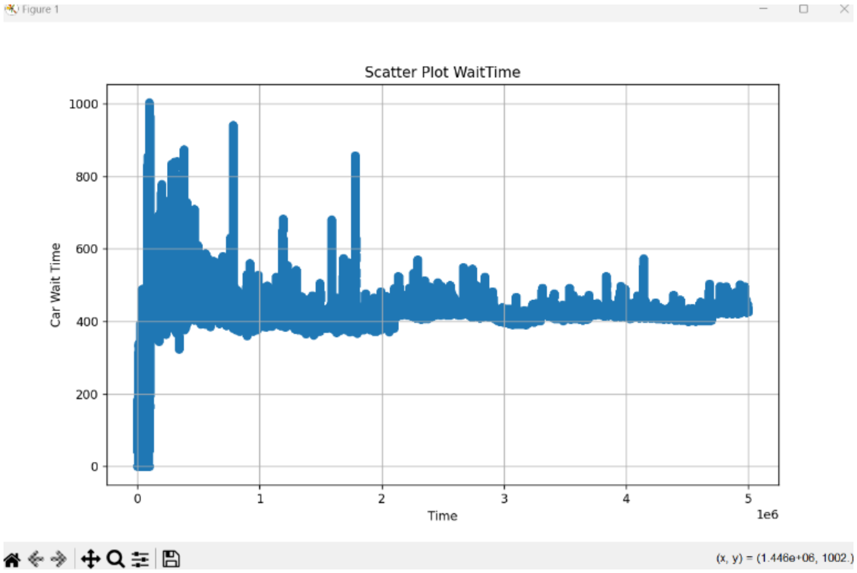


Figure 1
Scatter Plot WaitTime
Car Wait Time
0
200
400
600
800
1000
Time
0
1
2
3
4
5
1e6
(x, y) = (1.446e+06, 1002.)