CS 689 HW 3

Tianqi Li

Tuesday 30th October, 2018

1 Problem 1: Fixed time signal policy

For each type of phasing(protected, permissive-protected, split-NS, split-EW), it has fixed cycle time and fixed signals, the optimizing problem of total time loss is the optimization of the barrier time for the four signals, which can be represented as

$$a, b, c, d = \underset{(a,b,c,d):a+b+c+d=100}{\text{argmin}} L(a, b, c, d) \tag{1}$$

a,b,c,d is the time period for the four signals, L is the average Loss_time square.

To get the optimize solution, we try to list all combination of (a,b,c,d) and get the minimal L(·).

If by experiment, to collect all (a,b,c,d) combos, we assume $a, b, c, d > 2$, we can get 138415 different possible solutions, which will be computational heavy.

Add some reasoning constraints on optimization:

- through direction has main flow, so larger through may be possible; two left turn time is the same; ($a < b, c < d, a = c$)
- all signal's duration is larger than 15s. ($a, b, c, d > 15$)

With these two constraints, the possible number of (a,b,c,d) combos is 210, which becomes eligible for experiment. All result is in Table 1.

Table 1: Parameter for Modeling Delta Robot

Type	Minimal Loss_time ^{2}	Policy (a,b,c,d)
protected	14,452	(18,42,18,22)
protected-permissive	9,935	(15,46,15,24)
split-protect-NS	12,150	(15,50,15,20)
split-protect-EW	15,111	(20,29,20,31)

From the table 1, we see the solution of protected-permissive and split-protect-NS is on the boundary of our constraints, to improve the solution, I extend $a, b, c, d > 11$, and get the new solution for these two modes.

Table 2: Parameter for Modeling Delta Robot

Type	Minimal Loss_time ^{2}	Policy (a,b,c,d)
protected	14,452	(18,42,18,22)
protected-permissive	8,829	(11,40,11,38)
split-protect-NS	12,150	(15,50,15,20)
split-protect-EW	15,111	(20,29,20,31)

Please check `fixed_optimize.py` for this method.

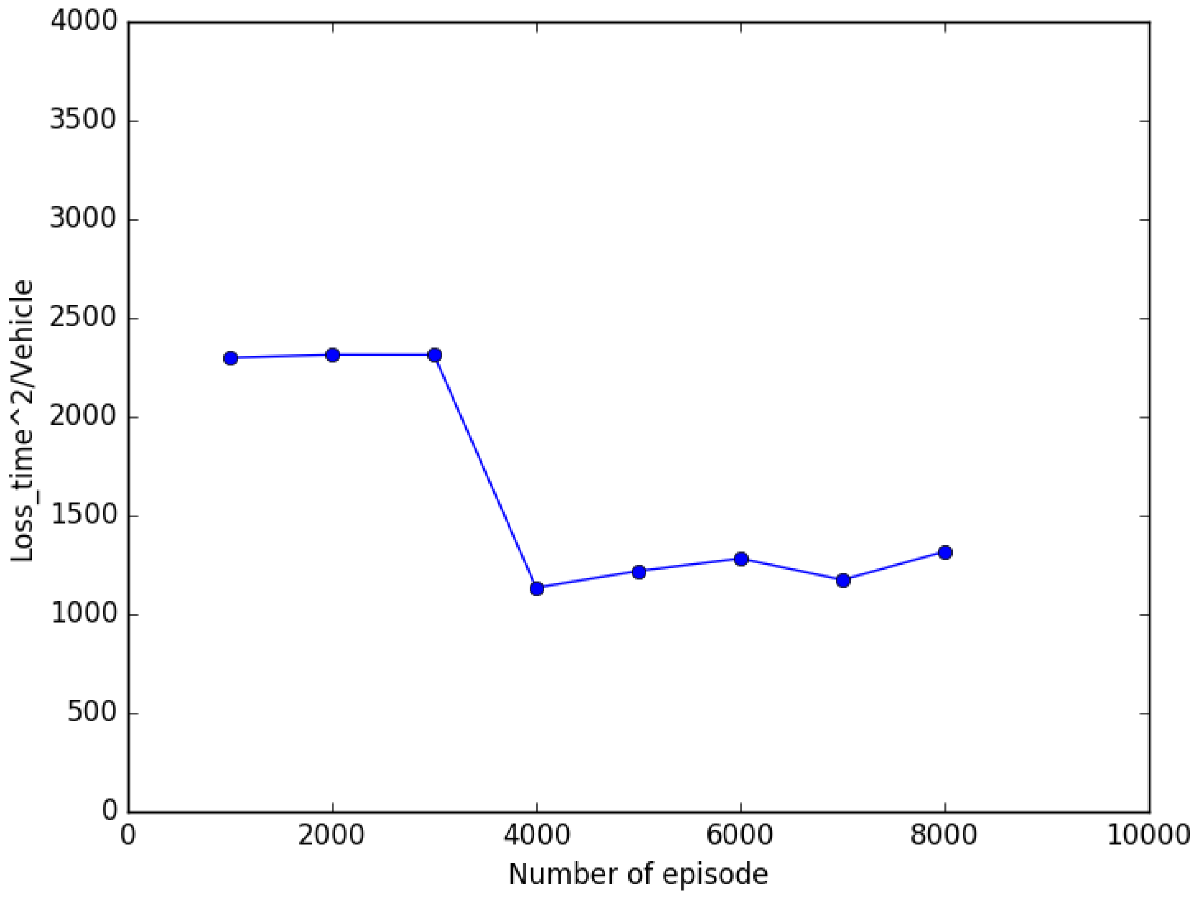

Figure 1: Performance of DQL during training

2 Problem 2: Actuated traffic law

In this section, sensors can be applied to analyze the traffic situation and make control policies. The pseudo control policy is in Algorithm 1.

With test in 9000s per run and 30 runs, the average Loss_time $^2 = 7,240$.

3 Problem 3: Deep Q-Learning traffic policy

Using ANN in learning the Q function, we are able to find out the best policy. In my training progress, I use (vehicle average speed of each lane, traffic light phase) as system state, action is 0/1 choice of changing the protected-permissive signalization. From Fig. 1, we can see tell the performance is 'learning' with less $Loss\_time^2$ while training. During the training process, several networks are tried with different layers and neurons, while I found this process is not robust without a set of 'correct' parameter in mini-batch size, epoch number and network set. More studies should focus on the resaon of the outcome difference.

The training document is `new_train.py`.

Algorithm 1 Actuated traffic law

```
Require: current light phase $l_t$, all lane's vehicle in waiting $f$ traci.areal.getJamLengthVehicle(), devided into two
  directions EW and NS, each directions have left and through.
  Threshold:
    • $a_1$ is the threshold to analyze if we need to switch from one direction to another direction;
    • $a_2$ is the threshold to analyze if we need to change from left/through to through/left

  detect traffic, get current green light direction t = (EW or NS), set the other direction is t'
  if $f(t, left) + f(t, through) < f(t', left) + f(t', through) - a_1$ then
    // Analyze if we need to change traffic direction, if the sum of flow in opposite direction is larger than sum of
    current green light direction, we need to switch green light direction
    if $f(t', left) > f(t', through)$ then
      make (opposite direction , left) green light
      return
    else if $f(t', left) < f(t', through)$ then
      make (opposite direction , through) green light
      return
    else
      make (opposite direction , through + left) green light
      return
    end if
  end if
  if $l_t$ is left + through then
    //if current flow dominates and don't need to change green light direction, check if need to change green light
    in current direction
    if $f(t, through) - f(t, left) > a_2$ then
      //too much vehicles waiting in through direction, change
      make (current direction , through) green light
      return
    else if $f(t, left) - f(t, through) > a_2$ then
      //too much vehicles waiting in through direction, change
      make (current direction , left) green light
      return
    else
      keep current state
      return
    end if
  else
    // current phase is single left or through
    if $abs(f(t, left) - f(t, through)) < a_2$ then
      //if vehicles in left and through direction are somehow in same level, switch to left + through to increase
      efficiency
      make (current direction , left + through) green light
      return
    else
      return
    end if
  end if
```