Intelligent transportation system

Chen Shang, Xinjun Li

University of Michigan

April 20, 2021

Motivation and challenges

- Traffic congestion is a very serious problems today, and it is also a complex problem which is hard to solve.
- Intelligence Transportation System is playing an important role in optimizing fuel efficiency, reducing delays and enhancing driving experience
- Current traffic light control policies are based on historical data and adapted in daily pattern, and the traffic light control systems based on real-time conditions is far more effective.
- Some technical challenges arise in analyzing real-time data.

Aim of our project

- Improve current traffic flow scenarios to reflect real world transportation problem better.
- Find traffic light control policies that will not cause any traffic accident.
- Optimize traffic light control policies based on some RL algorithms to minimize the waiting time.
- Visualization

Prior work and approach

- Traditional light control
  - Preprogramed cycles
  - No dependence on detailed states
- "Smart control"–classical MDP framework
  - State: # cars, # lights
  - Action: light switching(GG, RR, R-Y-G, G-Y-R)
  - Cost(negative reward): waiting time at the intersection
  - Bellman optimality equation:

$$Q^*(s,a) = R(s,a) + \sum_{s' \in S} T(s,a,s') \max_{a' \in A} Q^*(s',a'), \tag{1}$$

    where $Q^*(s,a)$, $R(s,a)$ and $\sum_{s' \in S} T(s,a,s') \max_{a' \in A} Q^*(s',a')$ represents quality of state-action pair $Q(s,a)$, current reward and future reward resp.
- Drawbacks:
  - Requires prior knowledge of state transition matrix (not model free)
  - State space grows exponentially with # intersections (not scalable)
- Our solution: Use deep-Q network

Single Road intersection

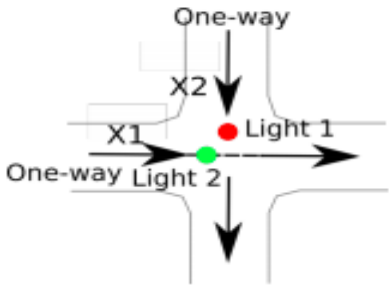

Figure: Single road intersection

The state $S(t)$ of the system at the beginning of time slot t may be described by the three-tuple $(X_1(t), X_2(t), Y(t))$ with $X_i(t)$ denoting the number of vehicles of traffic flow i waiting to cross the intersection and $Y(t) \in \{0, 1, 2, 3\}$ indicating the configuration of the traffic lights:

- "0": green light for direction 1 and hence red light for direction 2.
- "1": yellow light for direction 1 and hence red light for direction 2.
- "2": green light for direction 2 and hence red light for direction 1.
- "3": yellow light for direction 2 and hence red light for direction 1.

Single road intersection

Observe for each configuration k, we can either continue in the next time or switch to natural subsequent configuration $(k+1) \mod 4$.

This is determined by action $A(t) = \begin{cases} 0 & \text{for continue} \\ 1 & \text{for switch} \end{cases}$ as:

$$Y(t+1) = (Y(t) + A(t)) \mod 4 \tag{2}$$

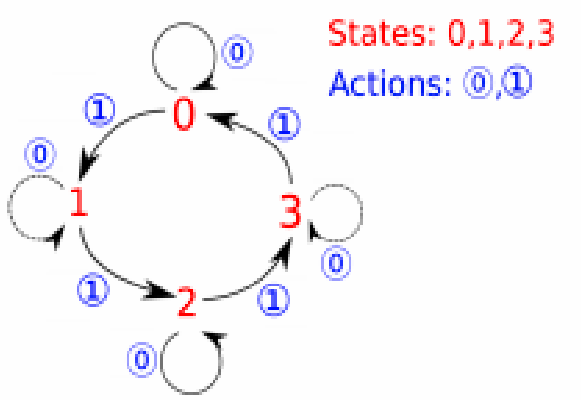

Figure: State Transition Graph

Single road intersection

The evolution of the quene state over time is governed by the recursion:

$$(X_1(t+1), X_2(t+1)) = (X_1(t)+C_1(t)-D_1(t), X_2(t)+C_2(t)-D_2(t)), \quad (3)$$

- $C_i(t)$: number of vehicles of traffic flow i appearing at the intersection during time slot t,
- $D_i(t)$: number of departing vehicles of traffic flow i crossing the intersection during time slot t.

Single road intersection

Assumption: If one of the two traffic flows is granted the green light, then exactly one waiting vehicle of that traffic flow, if any, will cross the intersection during that time slot, i.e.,

$$D_1(t) = \begin{cases} \min\{1, X_1(t)\} & \text{if } Y(t) = 0 \\ 0 & \text{if } Y(t) \neq 0 \end{cases} \tag{4}$$

$$D_2(t) = \begin{cases} \min\{1, X_2(t)\} & \text{if } Y(t) = 2 \\ 0 & \text{if } Y(t) \neq 2 \end{cases} \tag{5}$$

Linear road scenario

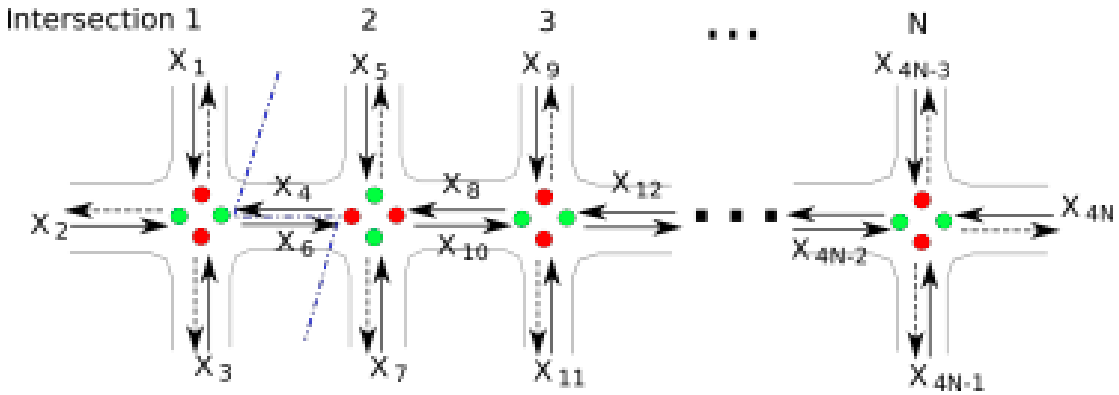

Figure: Linear bidirectional road network

The state $S(t)$ is described as $5N$ tuple – $(X_{n1}(t), X_{n2}(t), X_{n3}(t), X_{n4}(t); Y_n(t))_{n=1,\cdots,N}$ with directions 1 and 2 corresponding to East-West direction and North-South direction.

Linear road scenario

- Evolution of configuration

$$Y_n(t+1) = (Y_n(t) + A_n(t)) \mod 4 \tag{6}$$

- Evolution of states

$$X_{ni}(t+1) = X_{ni}(t) + C_{ni}(t) - D_{ni}(t) \tag{7}$$

  - $C_{ni}(t)$: the number of vehicles in direction i appearing at the nth intersection during time slot t
  - $D_{ni}(t)$: the number of vehicles crossing the nth intersection in direction i during time slot t
  - $C_{n+1,1}(t+u) = D_{n1}(t)$ and $C_{n3}(t+u) = D_{n+1,3}(t)$

Linear road scenario

Assumption:

$$D_{n1}(t) = \begin{cases} \min\{1, X_{n1}(t)\} & \text{if } Y_n(t) = 0 \\ 0 & \text{if } Y_n(t) \neq 0 \end{cases} \tag{8}$$

$$D_{n3}(t) = \begin{cases} \min\{1, X_{n3}(t)\} & \text{if } Y_n(t) = 0 \\ 0 & \text{if } Y_n(t) \neq 0 \end{cases} \tag{9}$$

Similarly for $D_{n2}(t)$ and $D_{n4}(t)$ depending on whether $Y_n(t) = 2$ or not.

Some Generalizations–pedestrians and vehicles intersection

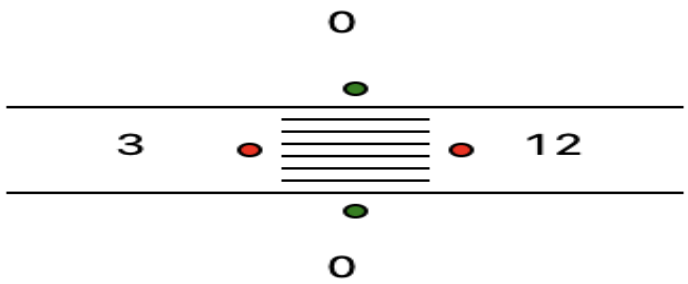

Figure: Pedestrian and vehicles

Some Generalizations–grid network case

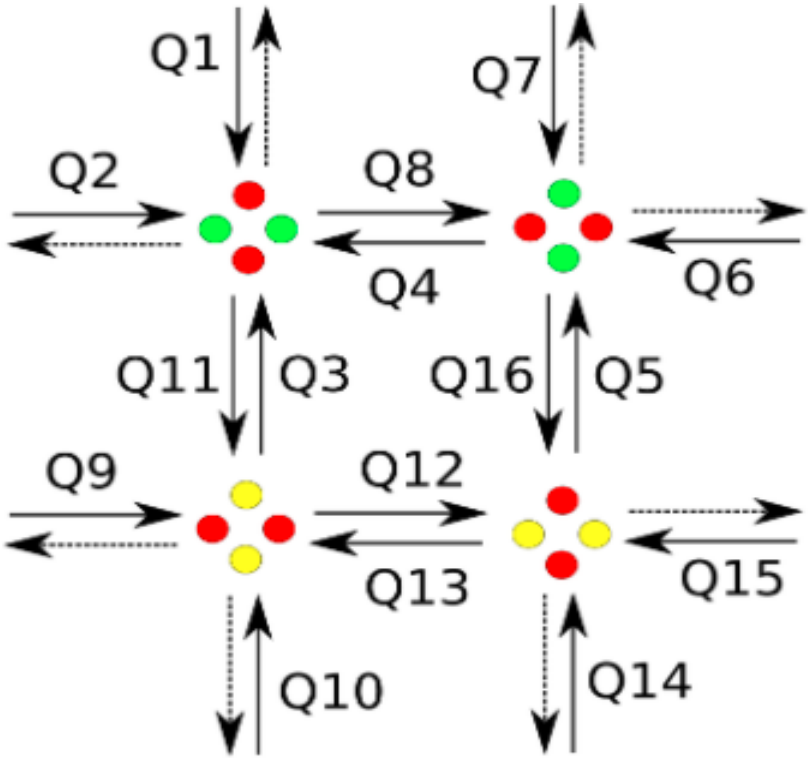

Figure: Grid network case

Optimization Goal

- Assume that the "congestion cost" in time slot t is a function $F(X(t))$
  - Single-intersection scenario: $X(t) = (X_1(t), X_2(t))$
  - Linear road case: $X(t) = (X_{n1}(t), X_{n2}(t), X_{n3}(t), X_{n4}(t))$
- Want to find a dynamic control policy to minimize the long-term expected discounted cost $\mathbb{E}[\sum_{t=1}^{\infty} \gamma^t F(X(t))]$, with $\gamma \in (0, 1)$ representing a discount factor.

Algorithm Design

- Denote $Q(s, a)$ as the maximum achievable expected discounted reward.
- Under optimal policy starting from state $s = (X, Y)$ when action a is taken,

$$Q(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s, s'; a) \max_{a' \in \mathcal{A}} Q(s', a') \tag{10}$$

$$= r(s, a) + \gamma \mathbb{E}[\max_{a' \in \mathcal{A}} Q(s', a')] \tag{11}$$

  - $\mathcal{S}$: state space and $\mathcal{A}$: action space
  - $r(s, a) = F(X)$ denotes the congestion cost in queue state X
  - $p(s, s'; a)$ denotes the transition probability from state s to state s' when action a is taken

Algorithm Design

Observe that the values $V(s) = \max_{a' \in \mathcal{A}}$ satisfy the Bellman optimality equations

$$V(s) = \max_{a' \in \mathcal{A}} \{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s, s'; a) V(s') \} \tag{12}$$

$$= \max_{a' \in \mathcal{A}} \{ r(s, a) + \gamma \mathbb{E}[V(s')] \} \tag{13}$$

Algorithm Design

Deep Q-network:

- System state S serve as input for both target and evaluation network
- Equation (10) provides basis for deriving the target Q-values at each time step
- Q-learning update for the neural approximator in the i^{th} iteration is calculated based on:

$$\ell(\theta) = \mathbb{E}_{s,a,r,s' \sim memory}[(r + \gamma max(q_{target}(s', a'; \theta_i')) - q_{eval}(s, a; \theta_i))^2] \tag{14}$$

  - r: reward in current step
  - s' and a': state and action in the next step
  - θ_i: parameters of the i^{th} evaluate Q-network
  - θ_i' : parameters of the t^{th} target Q-network

Algorithm Design

Deep Q-network basic idea:

- The DQN algorithm sample from and train on data collected in memory
- The online samples are stored in memory for further learning
- A warm-up period of k_0 time steps is applied before the learning operations are initiated,
- The evaluate network is updated with AdamOptimizer, gradient-descent and ϵ-greedy policy.

Algorithm Design

Algorithm 1 DQN for single intersection or linear road topology with N intersections

1: Initialize queue and control states: either $X_1, X_2 = 0; Y = 0$ [single intersection] or $X_{n1}, X_{n2}, X_{n3}, X_{n4} = 0; Y_n = 0$ for all $n = 1, \ldots, N$ [linear topology];
2: For steps $k = 1, \ldots, K$ do:
3: $s = [X_1, X_2; Y]$ [single intersection] or $s = [X_{11}, \ldots, X_{N4}; Y_1, \ldots Y_N]$ [linear topology];
4: Select $a^* = \arg\max_{a \in \mathcal{A}} q\_eval(s; a)$, using eval_net to evaluate the Q-value for each action;
5: Generate random variables C_1, C_2 [single intersection] or C_{11}, C_{N3} and C_{n2}, C_{n4} for all $n = 1, \ldots, N$;
6: Given a^*, determine new queue and control states X_1', X_2', Y' according to Eq. (1)-(3) [single intersection] or $X_{11}', \ldots, X_{N4}', Y_1', \ldots, Y_N'$ according to Eq. (4)-(6) [linear topology];
7: $r = -((X_1')^2 + (X_2')^2)$ [single intersection] or $r = -\sum_{n=1}^{N} \sum_{i=1}^{4} (X_{ni}')^2$ [linear topology];
8: $s' = [X_1', X_2'; Y']$ [single intersection] or $s' = [X_{11}', \ldots, X_{N4}'; Y_1', \ldots, Y_N']$ [linear topology];
9: Store transition $[s, a^*, r, s']$ in memory;
10: Perform learning operation if $k > k_0$:
11: Sample a minibatch of samples from memory;
12: Update target network: $\theta_i' = \theta_i$;
13: Calculate the target Q-value: $q\_target(s, a^*) = r + \gamma \max_{a' \in \mathcal{A}} q\_target(s', a')$;
14: Update evaluate network with gradient descent (using AdamOptimizer) and ϵ-greedy policy: $\text{Loss}(\theta_i) = \mathbb{E}[(q\_target - q\_eval)^2]$.

Single Road Intersection

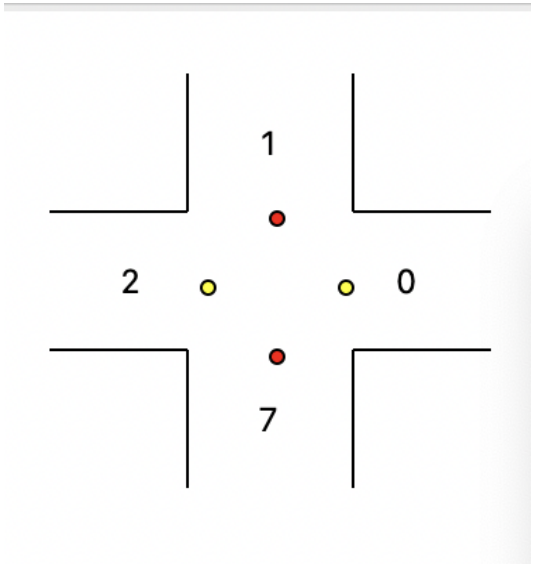

Figure: Traffic flow on one intersection

Single Road Intersection

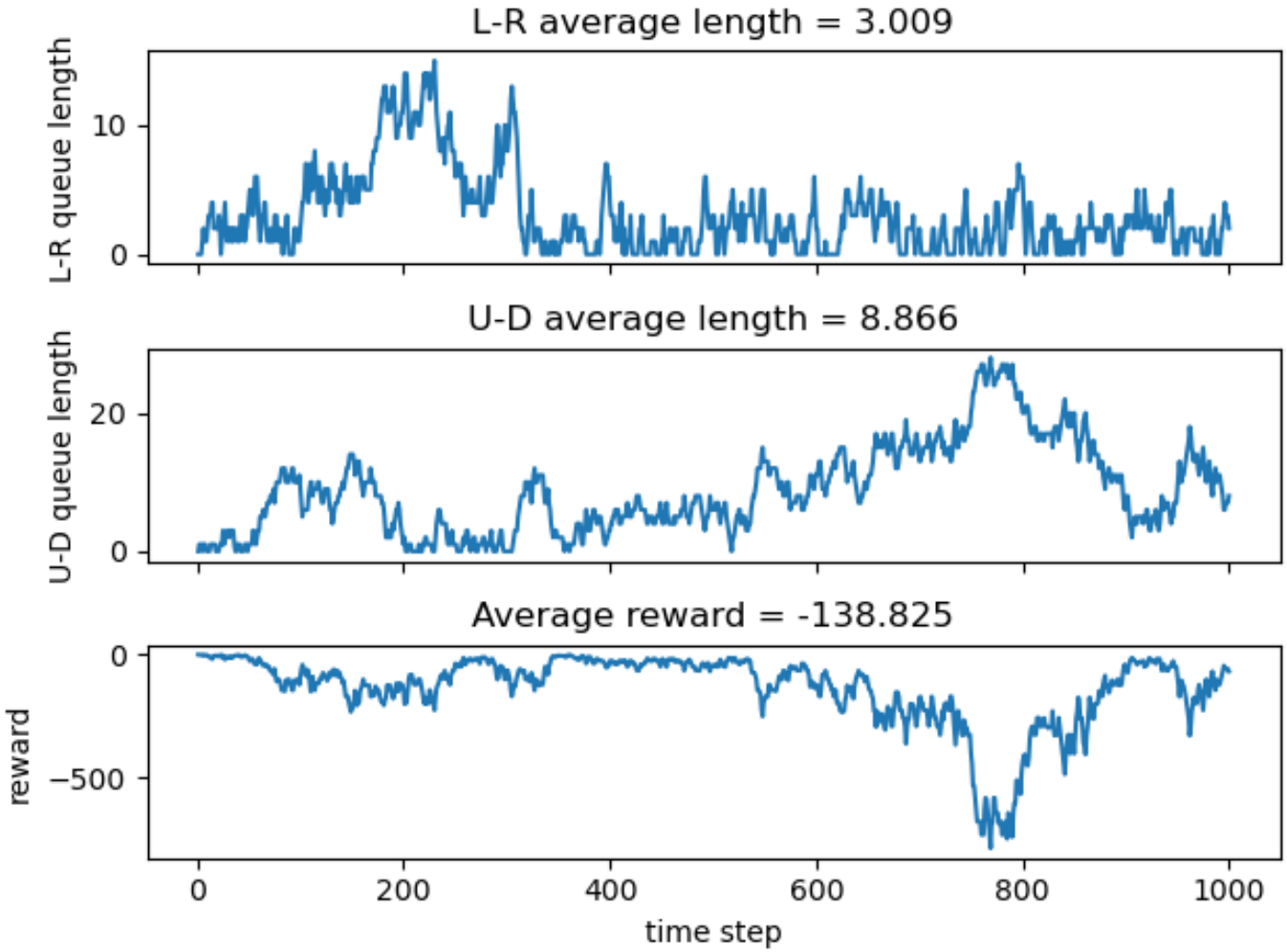

Figure: Performance of DQN on single intersection

Crosswalk

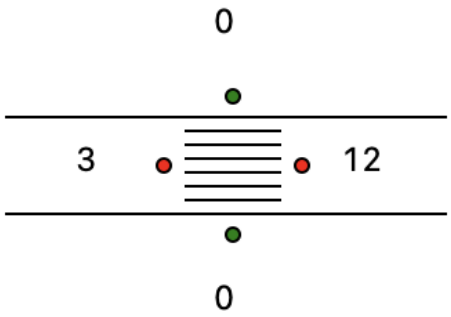

Figure: Traffic flow on a crosswalk

Crosswalk

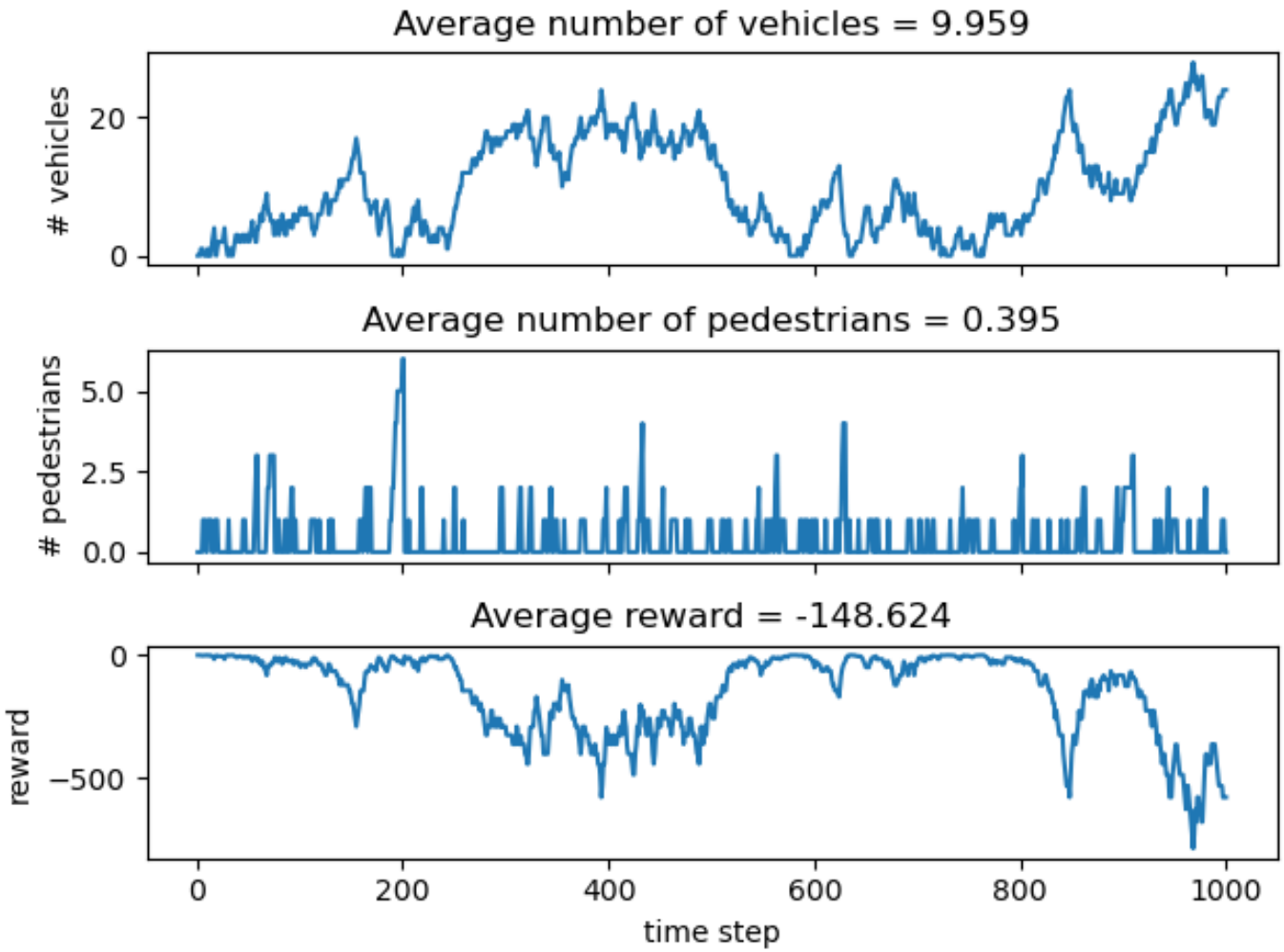

Figure: Performance of DQN on a crosswalk

Linear road scenario

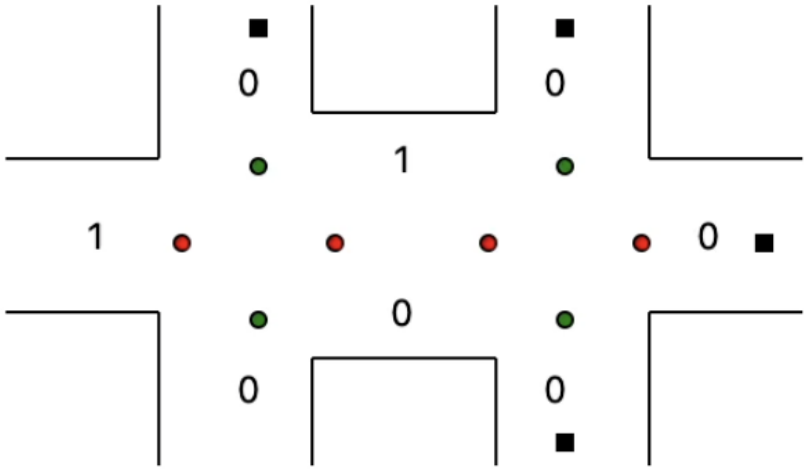

Figure: Traffic flow on linear bidirectional road network (with $N = 2$)

Linear road scenario

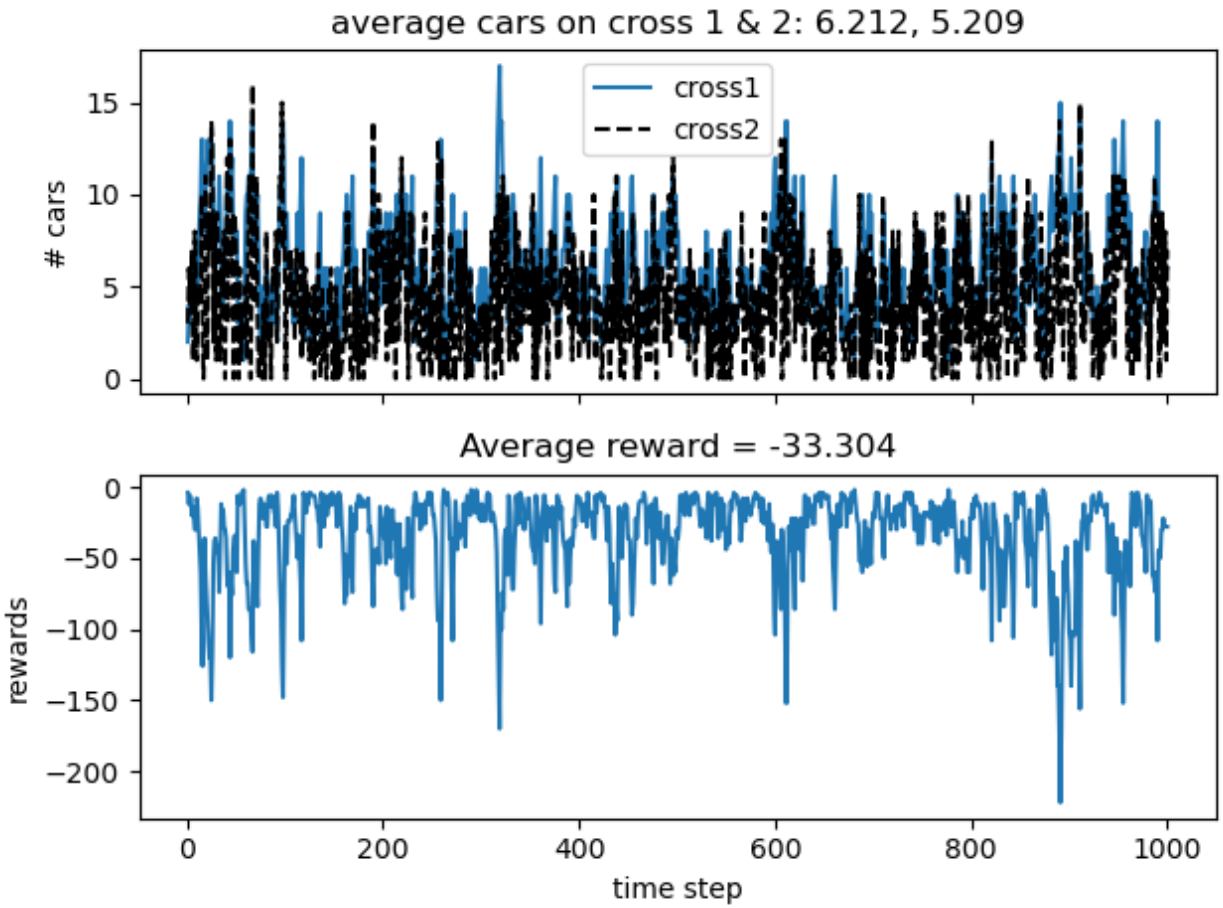

Figure: Performance of DQN on linear bidirectional road network (with $N = 2$)

Grid Network Intersections

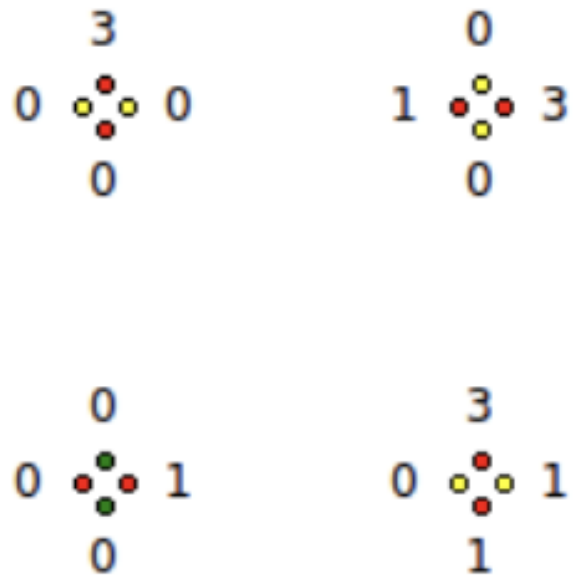

Figure: Traffic flow on Grid Network Intersections (2×2)

Grid Network Intersections

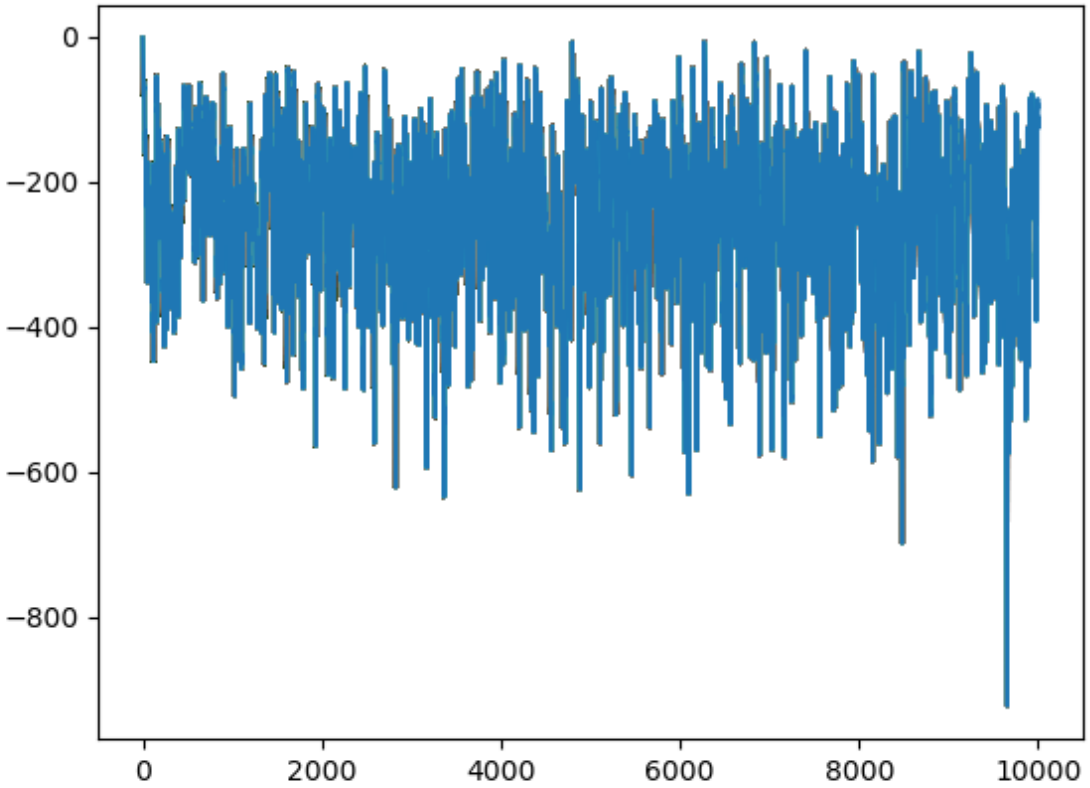

Figure: Performance of DQN on Grid Network Intersections (2×2)

Conclusion

We have explored the scope for Deep Q-Networks (DQN) to optimize real-time traffic light control policies in emerging large-scale Intelligent Transportation Systems.
With Deep Q-Networks, the traffic light system on the intersections can make the number of vehicles remains low.
Deep Q-Networks also do will on handling the "green-wave" situation, where the number of cars on a road grow unexpectedly. DQN can react to "green-wave" and reduce the number of cars on a certain road.

Future Works

In the future, we can explore the application of Reinforcement Learning on Intelligent Transportation Systems by considering more scenarios such as intersection turning lanes, vehicles of different shapes and sizes, etc. We can also apply multi-agent on an intersection to find out some strategies.

Reference

- Xiaoyang-Liu, Zihan Ding, Sem Borst, and Andrew Walid. Deep Reinforcement learning for Intelligent Transportation Systems. October 2018.
- J. Zhang, F. Wang, K.Wang, W. Lin, X.Xu, and C.Chen. Data-Driven Intelligent Transportation Systems: A survey. *IEEE Transactions on intelligent Transportation Systems* 12(4): 1624-1639. Deceber 2011
- Rusheng Zhang, Akihiro Ishikawa, Wenli Wang, Benjamin Striner, and Ozan Tonguz. Using Reinforcement Learning with Partial Vehicle Detection for Intelligent Traffic Signal Control. February 2020.