Mixed Autonomy Reducing Congestion with Reinforcement Learning

Toshinori Kitamura

June 8, 2019

Abstract

This research demonstrates the potential of multi-agent reinforcement learning to reduce traffic congestion. The main goal is to solve the remaining problems of [1], which uses only a single agent to control all the RL cars on the road. If we use a single agent, the number of the cars is not flexible, and it is not safe from the view of cyber security. Instead, this research focus on multi-agent method and solve the problems by modifying state space and reward function. Since the reward design is difficult for this task, an imitation learning algorithm GAIL is applied to the traffic control as another experiment.

I. Introduction

This paper shows an application of reinforcement learning for congestion reduction by mixed-autonomy, where both automated and human-driven vehicles present on traffic[2]. Some studies demonstrate that even a small number of vehicles in congested traffic can reduce huge amount of fuel consumption[2].
Since 28% of energy consumption in the U.S. is due to the transportation[3], and congestion can increase the fuel consumption[4], reducing traffic congestion saves large amount of energy. To control the system-level traffic velocity, both model-based and model-free methods have been studied so far. The model-based methods give analytical solutions to maximize the traffic efficiency, but the situation it can be utilized is very limited. For example, the model-based controllers "Follower Stopper" and "PI with Saturation" [5] can reduce the congestion in a ring road, but it doesn't give the optimal solution in the complex environments such as bottleneck and figure-eight(see Fig.1). Due to the complex model of the real world traffic, the model-based methods may not work in many situations.

On the other hand, model-free reinforcement learning(RL) has a potential to solve complex problems. For example, [6] trains an agents which solves Atari games at superhuman levels, and [7] demonstrates that a RL agent outperforms a human Go player. [8] applies some state-of-the-art RL algorithms to complex traffic shown in Fig. 1, and most of the methods achieve better score than the situation with only human driven car.

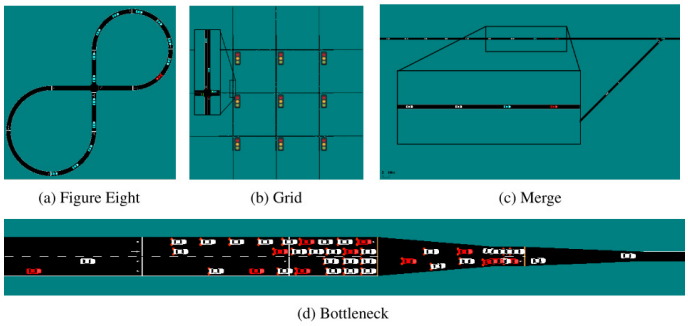
This research extends that previous work[1] and develops more practical agent using multi-agents method. A remaining problem in [1] is that it only uses one agent to control multiple cars. It's not flexible when the number of the car changes. Moreover, from the view of cyber security, one master node controlling the acceleration of multiple cars is dangerous. To train the agents, a policy gradient algorithm, Proximal Policy Optimization(PPO)[9] is used. The traffic simulator used in this work is Flow[10]. It contains some benchmarks of practical traffic situations shown as 1, and it is easy to apply reinforcement learning agents to Flow.

Finally, this paper shows the performance of imitation learning for the traffic control task. Generally, it is difficult to design a reward function for reinforcement learning. One of the solutions for reward design problem is to learn from an expert agent using imitation learning. To apply imitation learning, Generative Adversarial Imitation Learning(GAIL)[11] is used in this research.

II. Background

i. Reinforcement Learning

RL is an optimization algorithm in Markov decision process(MDP), defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma, T)$. Here, $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{P}$ is a state transition probability function, r is a reward function, ρ_0 is initial state distribution, γ is a discount factor, and T is a time horizon.



(a) Figure Eight (b) Grid (c) Merge

(d) Bottleneck

Figure 1: *Complex traffic environments in "flow" (cited from [8])*

In RL algorithms, agents interact the environment and try to learn a parameterized policy π_θ which maximize the sum of the discounted reward $\sum_{i=0}^{T} \gamma^i r_i$, i.e. $\pi_\theta = \arg\max_{\pi_\theta} \sum_{i=0}^{T} \gamma^i r_i$. The rewards are sampled from the trajectory $\tau = (s_0, a_0, \ldots, a_{T-1}, s_T)$. The initial states, actions and the next states are sampled as $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi_\theta(a_t)$, and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t, a_t)$

The plain policy gradient(PG) method updates the parameter using following gradient.

$$\nabla J(\pi_\theta) = E\left[\nabla_\theta \log \pi_\theta(a|s) Q_\pi(s,a)\right] \tag{1}$$

Q_π is an action value function. The problem of plain PG is its instability. Intuitively, the instability is from the term $\nabla_\theta \log \pi_\theta(a|s)$. As the training goes, the policy is getting deterministic, and the gradient of the policy becomes large around its mean value. PPO solves the problem by restricting the update in parameter space, and achieves stabler RL training.

ii. Imitation Learning

Designing reward function is difficult problem in RL. Instead, IL trains the agents from the trajectory generated by an expert. [11] shows that the IL problem is same as a problem finding a policy whose occupancy measure ρ_π matches the one of the expert. Here, occupancy measure is the distribution of state and action pair that the agent encounters during an episode. GAIL, which used in this research, exploits this feature. The structure of GAIL is similar to the structure of generative adversarial network(GAN)[12]. GAIL has a discriminator which tries to tell whether the trajectory comes from the expert or the agent trained by GAIL. As the training goes, the trajectory generated by the GAIL is getting similar to the one from the expert.

III. Experiments

i. Experiment details

To evaluate the RL performance, the merge network(see Fig. 1(c)) is chosen as a benchmark. This network is given by the benchmark paper[8]. In the merge network, there is a highway and an intersection. The intersection can be considered as disturbance to the highway. Due to the disturbance, when the inflow rate reaches to a certain level, stop-and-go wave occurs at the intersection, and it propagates from right to left. The goal of this task is to reduce the stop-and-go wave by controlling 10% of cars on the road using RL.

The details of the environment is as follows. To apply multi-agents RL, the states, actions, reward are changed from the benchmark.

- States: In this environment, the RL cars can communicate to the leading RL car if it exists, i.e. the state s consists of two parts, the information of the own car$i$ and the information of the leading RL cari_{lead}. The information, i or i_{lead}, consists of 6 elements: speeds, bumper-to-bumper distance of the leading and following cars, and the distance to the intersection, i.e. $i := \left(v_{i,\text{ lead}}, v_{i,\text{ lag}}, h_{i,\text{ lead}}, h_{i,\text{ lag}}, v_i, d\right)$. The state s consists of two information, i.e. $s := (i, i_{\text{ lead}})$.
- Actions: Action a is the acceleration to the RL car. The action value is bounded by the minimum and maximum acceleration. Then, the action a is $a \in \mathbb{R}_{[a_{\min}, a_{\max}]}$
- Reward: The reward function is based on the original benchmark, but slightly changed to support multi-agent.

  $$r = \|v_{\text{des}}\| - \|v_{\text{des}} - v(t)\| - \alpha \max\left[h_{\max} - h_i(t), 0\right] \tag{2}$$

  Eq. 2 is the original reward function given by the previous work[1]. To support multi-agent, a penalty term is added to the reward function, i.e. the reward function is as follows.

  $$r = \|v_{\text{des}}\| - \|v_{\text{des}} - v(t)\| - \alpha \max\left[h_{\max} - h_i(t), 0\right] - c \tag{3}$$

  Compared to the original reward function, $c \in \mathcal{R}$ is subtracted from the reward function. This penalty term makes the agent get out from the highway as

fast as it can.
In this research, the values of the parameters are $v_{\text{des}} = 25$, $h_{\text{max}} = 10$, $c = 1.0$

- Traffic information:
  - Inflow rate of the highway: 2000 veh/hr
  - Inflow rate of the disturbance: 100 veh/hr
  - RL car ratio: 10%

The detail of the training is as follows. The agent of the GAIL is updated by PPO.

- PPO model:
  - hidden layers: (128, 64, 32)
  - GAE[13] lambda: 0.97
  - learning rate: 5×10^{-4}
- GAIL discriminator:
  - hidden layers: (128, 128)
  - learning rate: 0.005
- training:
  - number of cpus: 20
  - training iteration(PPO): 50
  - training iteration(GAIL): 150

ii. Results

Fig. 2 shows how much congestion occurs on the highway. In the situation without any RL cars, the congestion occurred at the intersection and it propagates as a stop-and-go wave. On the other hand, the PPO and GAIL agents successfully reduce the congestion.

Fig. 3 and 4 show the velocity and the outflow rate of the highway. Without RL, the mean velocity decreases and converges about 8 m/s around 300 steps. On the other hand, the trained PPO and GAIL keep the speed around 15 m/s. The PPO agent increases the velocity of the traffic about 40% with almost the same outflow rate(see Table. 1).

The GAIL agent uses the trained PPO agent as the expert. Even though the GAIL agent is not trained on the reward function3, the performance and the behavior of the GAIL agent is pretty similar to that of the PPO agent. The GAIL agent success fully reduce the congestion(Fig. 2) and increase the velocity(Fig. 4).

Since the inflow rate is set as 2000 veh/hr, the outflow rate should be 2000 veh/hr. However, the measured outflow rate is almost 75% of the inflow rate. This phenomenon is due to the system of the simulator. The simulator stops inserting new cars if cars exist at the start point of the highway. The agent learns how to reduce the inflow rate by waiting at the start point. This reducing traffic effect is similar to the ramp metering on the highway.

	No RL	PPO	GAIL
Mean Velocity (m/s)	10.1	14.0	13.5
Outflow rate (veh/hr)	1421	1497	1551

Table 1: *Mean velocity and outflow rate by one episode. PPO and GAIL achieve high velocity while keeping almost the same outflow rate.*

IV. Discussion

This research solves the problems of the previous work[1]. The two problems of the previous work is, the limitation of the number of the RL cars and the vulnerability of the master node. Compared to [1], multi-agent doesn't require one master node to control all the vehicles, and each RL cars works independently. Therefore, the multi-agent one is safer than the one master node and more flexible. However, there are still many space to improve. The future works are as follows.

i. Model generalization

The agent is trained only on the merge network, so it is not guaranteed that it works on the other environments(e.g. figure eight, grid, bottleneck (see Fig. 1)). A future work is to train a more general agent which works in a variety of environments. As for the environments, there might be some problems on the original flow benchmarks. As mentioned in III, agents can lean how to reduce the outflow rate in the merge network. If the agent stop at the start point on the real traffic, congestion can happen at the start point. To measure the performance of the agent more accurately, better benchmark which prevents this phenomenon is necessary.

ii. For the real world application

The results of the GAIL shows the potential to reproduce an expert performance from trajectories. IL can solve the

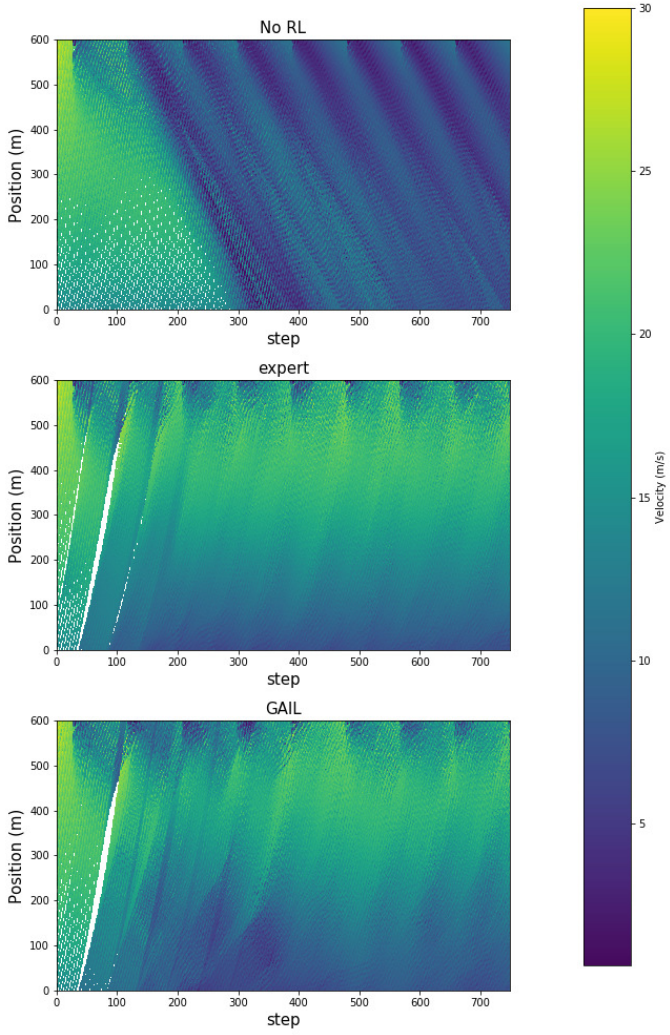


Figure 2: *Heat map of congestion level on the road. The results are averaged by 50 episodes. From top, the figures are the results of "No RL", "PPO" and "GAIL". If there are slow cars on the road, it appears as blue part. The "No RL" result has stop-and-go waves as blue lines propagating from top left to bottom right, but the other results doesn't have that.*

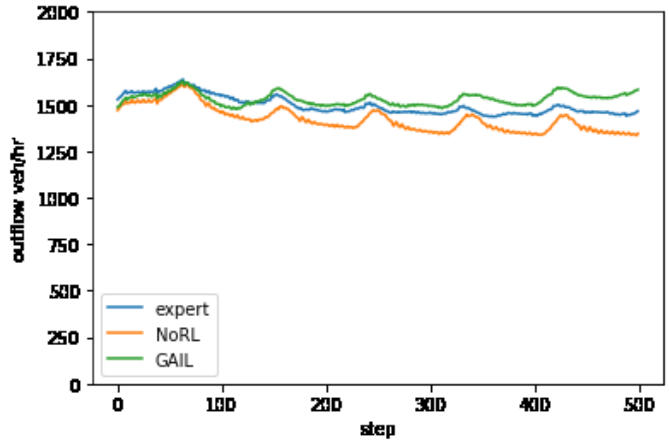


Figure 3: *Outflow rate during one episode. The results are averaged by 50 episodes. These three results keep almost the same outflow rate, around 1500 veh/hr.*

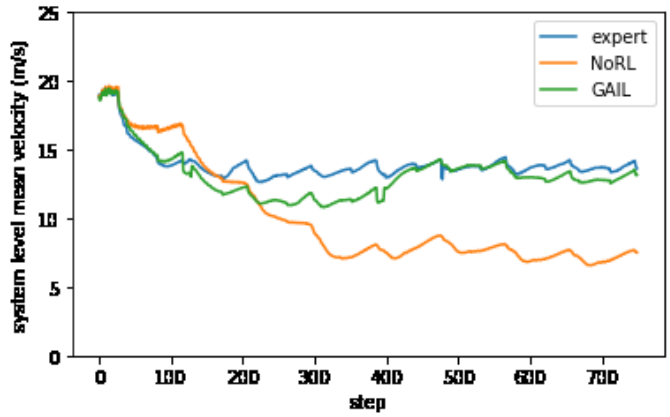


Figure 4: *Mean velocity of the traffic. The results are averaged by 50 episodes. Compared to the result of "No RL", GAIL and PPO keep high velocity during the episode.*

reward function problem, and it is useful especially in the real world tasks. In the real world, it is easy to obtain the trajectories using some sensors. The obtained trajectories can be used as an expert, and from which it is possible to reconstruct an agent imitating the behavior. The problem of GAIL is its sample inefficiency. GAIL consumes about 1 million samples to train an agent, and it requires 20 CPUs and 150 iteration of simulation. To find more sample efficient algorithms is a future work.

iii. Better algorithms

Another problem of GAIL for this task is the biased trajectory from the expert. The expert agent keeps a certain length of bumper-to-bumper distance and accelerates when there is a enough distance. After a few seconds from time $t = 0$, all the RL cars have enough distance and they converge to a stable traffic where they can drive with constant speed(free-flow). The problem is that the most of the part of the expert trajectory is from the free-flow traffic. Therefore, the number of samples from congested traffic is less than that of the free-flow traffic, so the expert sample is biased. It might make the training slower. DQN[6] has a similar problem, but it is solved by prioritized experience replay[14]. The idea is to use the sample which have huge effects on the training. This idea can be used in the IL work and solve the problem. To apply such less biased algorithm to the traffic control is another future work.

One solution for sample efficient algorithm is model-based RL. [15], [16] apply Model Predictive Control(MPC) to the RL training part, and successfully reduce the number of samples to train an agent controlling the real world robot. Although the model-predictive control is powerful as a model-based method, some changes are required if we want to apply it for the mixed-autonomy task. The problem is due to the fact that the reward function3 cannot be calculated from the state space, since it contains the mean velocity of the all the cars. Most of the model-based RL predicts the next state s_{t+1} from the current state s_t and action a_t pair, and calculate the reward r_{t+1} from the pair. Some predictions of the reward function or the other approach are necessary since the r_{t+1} cannot be calculated from the pair in this task.

A promising approach is using inverse RL[17]. Inverse RL reconstructs the reward function from the trajectory of a policy. Since the generated reward function by inverse RL is estimated from the state and action pair, it is possible to use the reward to the MPC. In short, the algorithm proposal consists of two steps, 1. reconstruct the reward function $r_t(s_t, a_t)$ from an expert, 2. use MPC and do model-based RL to train the agent.

References

[1] Abdul Rahman Kreidieh, Cathy Wu, and Alexandre M. Bayen. Dissipating stop-and-go waves in closed and open networks via deep reinforcement learning. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018-Novem:1475–1480, 2018.

[2] Cathy Wu. *Learning and Optimization for Mixed Autonomy Systems - A Mobility Context*. PhD thesis, EECS Department, University of California, Berkeley, Sep 2018.

[3] U.S. Energy Information Administration. Monthly Energy Review. Technical report, 2017.

[4] Martin Treiber, Arne Kesting, Christian Thiemann, M Treiber, A Kesting, and C Thiemann. How Much does Traffic Congestion Increase Fuel Consumption and Emissions? Applying a Fuel Consumption Model to the NGSIM Trajectory Data. *87th Annual Meeting of the Transportation Research Board, Washington, DC.*, 2008.

[5] Raphael E. Stern, Shumo Cui, Maria Laura Delle Monache, Rahul Bhadani, Matt Bunting, Miles Churchill, Nathaniel Hamilton, R'mani Haulcy, Hannah Pohlmann, Fangyu Wu, Benedetto Piccoli, Benjamin Seibold, Jonathan Sprinkle, and Daniel B. Work. Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments. *Transportation Research Part C: Emerging Technologies*, 89:205–221, may 2018.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. pages 1–9, 2013.

[7] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel,

and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, oct 2017.

[8] Eugene Vinitsky, Aboudy Kreidieh, Luc Le Flem, Nishant Kheterpal, Kathy Jang, Cathy Wu, Fangyu Wu, Richard Liaw, Eric Liang, and Alexandre M Bayen. Benchmarks for reinforcement learning in mixed-autonomy traffic. *Proceedings of The 2nd Conference on Robot Learning*, 87(CoRL):399–409, 2018.

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. 2017.

[10] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: Architecture and Benchmarking for Reinforcement Learning in Traffic Control. pages 1–16, 2017.

[11] Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. Technical report.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. jun 2014.

[13] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. 2015.

[14] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. 2015.

[15] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. aug 2017.

[16] Anusha Nagabandi, Guangzhao Yang, Thomas Asmar, Ravi Pandya, Gregory Kahn, Sergey Levine, and Ronald S Fearing. Learning Image-Conditioned Dynamics Models for Control of Under-actuated Legged Millirobots. Technical report.

[17] Andrew Y. Ng, Andrew Y. Ng, and Stuart Russell. Algorithms for Inverse Reinforcement Learning. *IN PROC. 17TH INTERNATIONAL CONF. ON MACHINE LEARNING*, pages 663—-670, 2000.