Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Platoon control of connected autonomous vehicles: A distributed reinforcement learning method by consensus

Bo Liu * **Zhengtao Ding** * **Chen Lv** **

* *Department of Electrical and Electronic Engineering, University of Manchester, Manchester M13 9PL, UK (emails: bo.liu-2@manchester.ac.uk, zhengtao.ding@manchester.ac.uk)*
** *C. Lv is with School of Mechanical and Aerospace Engineering, Nanyang Technological University, 639798, Singapore (E-mail: lyuchen@ntu.edu.sg)*

Abstract: This paper proposes a distributed reinforcement learning method based on deep Q-network and the consensus algorithm to deal with the multi-vehicle platoon control problem, which contains the two processes of local training and global consensus. The platooning problem is decomposed into many single-vehicle tasks based on deep Q-network, where each vehicle accumulates its experience data samples by interacting with its front and back vehicles. After initialization, all vehicles' Q-networks are first locally optimized based on their own experience simultaneously. The consensus algorithm is then used to make all vehicles in a decentralized platoon approach each other, where the communication is only required among directly connected vehicles. At last, the simulation study shows that the Q-networks of all vehicles reach consensus first and then converge to the optimum in union using the proposed distributed deep Q-networks algorithm, and all vehicles learn to form the required platoon and move forward with a roughly equal separation.

Keywords: Distributed training, Reinforcement learning, Platoon, Consensus

1. INTRODUCTION

With the rapid growth of vehicle ownership in the past few decades, the transportation infrastructure is under increasing pressure, which is more likely to cause road congestion and traffic accident. Traffic efficiency and safety, therefore, become an important demand in road transportation, especially for connected vehicles in future smart cities Lu et al. (2014); Zhang et al. (2018). The platooning of autonomous connected vehicles is a promising solution to improve traffic capacity and transportation efficiency, the main objective of which is to control a fleet of vehicles to keep a given separation distance while moving forward.

In recent years, researchers have proposed many advanced control strategies based on the multi-agent consensus framework for platoon control Lu et al. (2014); Zhang and Orosz (2016). Zheng et al. (2016) designed the distributed model predictive control algorithm for vehicle platoons over unidirectional graphs, and an equality-based terminal constraint is used to enforce all vehicles to converge to each other in the predictive horizon, which is proved to be asymptotically stable. Ploeg et al. (2013) applied the H-infinity controller to cooperative adaptive cruise control of vehicles with linear dynamics to improve road throughput, and the proposed method is experimentally validated using a platoon of three passenger vehicles to illustrate its practical feasibility. The H-infinity controller for platooning is extended to undirected topologies with robustness analysis in Zheng et al. (2017).

However, these methods require a specific vehicle dynamics model, and the longitudinal dynamics of a vehicle is inherently nonlinear and quite complex Liang et al. (2019), which is usually simplified to a linear vehicle model based on some assumptions Li et al. (2015). This simplified linear model is suitable for theoretic analysis but may not be applicable for real-life practice. In this paper, reinforcement learning is employed to achieve the control strategy for platooning without the requirement for the dynamics model of the vehicles. Deep Q-Network (DQN) Mnih et al. (2015) is a representative reinforcement learning method, which can reach human-level performance on many video games Mnih et al. (2013) with the combination of Q-learning and neural networks. DQN is designed for the single-agent architecture, which is not directly appropriate for this multi-vehicle platooning problem. This platooning of connected vehicles is a kind of multi-agent system problem Knorn et al. (2015) and can be solved more efficiently in a parallel manner.

Nair et al. (2015) proposed to train the agents using the reinforcement learning method in a distributed manner, where each agent interacts with its copy of the environment to fill the experience relay memory and computes the gradient of the DQN loss. These gradients are asynchronously sent to a central parameter server to update the central copy of the model, and the updated parameters are then sent back to each agent at fixed steps. Mnih et al. (2016) provided a parallel framework for both on-policy and off-policy reinforcement learning algorithms, where

multiple agents interact with the same environment and optimize the neural network controllers asynchronously to decorrelate all agents' experience samples into a more stationary process, which shows better results and requires less computational resources. Horgan et al. (2018) proposed a distributed architecture for deep reinforcement learning, which decouples acting from learning, that is, multiple actors interact with their copies of the environment but share the same neural network and experience replay memory, while the single learner updates the neural network based on the shared memory using prioritized experience replay Schaul et al. (2015).

The above parallel reinforcement learning frameworks are all based on the parameter server framework, where a central node is required to collect and summarizes other agents' information. This framework would lead to two main problems; one is the heavy communication burden on the central node, the other one is the collection or sharing of the experience samples may not be available because of the privacy-related issues and the limitation of communication bandwidth. In this paper, we propose a distributed training framework for deep Q-networks to deal with the multi-vehicle platooning problem, and the main contributions can be summarized as follows:

1) The dynamics model of a vehicle is not required for this platoon problem, and each vehicle only need to obtain the positions of its front and back vehicles for modeling and decision;

2) The proposed method is suitable for the decentralized platoon without a central node, which avoids the possible communication traffic jam;

3) The Q-network parameters of all vehicles, other than their experience data samples, are shared during the modeling process, which greatly benefits the privacy protection and alleviate the communication burden.

2. PROBLEM FORMULATION AND METHOD

2.1 Platooning of connected vehicles

We consider a platoon of automated connected vehicles on a road, including a leading vehicle and multiple following vehicles, the aim of which is to form a vehicle fleet with the same separation distance among all vehicles while moving forward. As mentioned before, the model-based control strategies are not suitable for this platooning problem in practice, because the detailed nonlinear vehicle dynamics are quite complex and hard to obtain. The reinforcement learning method is thus considered, which does not require the dynamical model of the vehicles.

In this platoon scenario, all the follower vehicles form a multi-agent system (MAS) Yang et al. (2016) to complete the task of platooning cooperatively. Furthermore, each vehicle can be taken as an independent vehicle with the same task of reaching the midpoint of its front and back vehicles. It is obvious that the vehicles will form a platoon if all the vehicles reach the midpoint of their front and back vehicles. In this way, this multi-vehicle platooning problem is decomposed into many similar single-vehicle reinforcement learning problems. That is, each vehicle takes its front and back vehicles as the environment to learn the task of keeping the same distance to its front and back vehicles based on its experience by interacting with them. More specifically, the vehicle gets a reward or penalty from the environment based on its action which transfers the vehicle from a state to another state. After repeating the process for some learning iterations, the vehicle learns to take appropriate actions to achieve the desired state with high reward.

It is reasonable that the learning process of a vehicle will speed up if the vehicle can learn from other vehicles' experience as all the vehicles are facing the same reinforcement learning problem. A simple and direct idea is to share their experience data samples among all vehicles in the platoon to learn faster. However, the collection or sharing of the experience samples may not be available. The reason is not only about the limitation of communication bandwidth for the platoon but also the concerns on privacy-related issues, especially for the driving behavior data.

Therefore, the problem is that it is better for each vehicle to learn from all vehicles' experience data, not just its local data, but it is infeasible for a vehicle to share its data with any other vehicles or a server center.

2.2 Deep Q-network

In the single-vehicle reinforcement learning scenario, the vehicle accumulates its experience samples by interacting with its front and back vehicles, to learn the optimal policy for keeping the same separation distance to its front and back vehicles in discrete steps.

An experience data sample of the vehicle at step t is defined by a tuple $(s_t, a_t, r_{t+1}, s_{t+1})$. More specifically, s_t is the current state in the environment, a_t is the selected action from the possible actions according to its policy π, while s_{t+1} and r_{t+1} are the next state and the received reward, respectively. The true value of the action a_t in the state s_t is $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$, which is the total accumulated reward from step t, with $\gamma \in (0, 1]$ being the discount factor to trade off the immediate reward and the later rewards.

As the true value of the action a in state s is hard to compute during the actual learning process, a main work for reinforcement learning is to estimate the value of all actions at different states under the policy π. that is

$$Q_\pi(s, a) = \mathbb{E}[R_t | s_t = s, a_t = a]. \tag{1}$$

The optimal value is $Q_*(s, a) = \max_\pi Q_\pi(s, a)$, based on which, the agent can easily obtain the optimal strategy by choosing the action with maximal value in each state. Q-learning is a popular method for estimating the optimal action values by updating a table recording all the $Q(s, a)$, while most problems in practice contain too many or infinite states, and this makes it impossible to learn all action values in all states separately.

The neural network is a good approximate method to represent a parameterized action value function $Q(s, a; \theta)$ because of its excellent fitting capacity Gurney (2014). The parameter θ of the value function is updated by minimizing a loss function at each step t using gradient descent methods, which is

$$l(e,\theta) = \frac{1}{2}[y - Q(s_t, a_t; \theta)]^2, \tag{2}$$

where $y = r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta)$ is the target action value of the $Q(s_t, a_t; \theta)$, and the tuple $e = (s_t, a_t, r_{t+1}, s_{t+1})$ is an experience sample for minimizing the loss function $l(e, \theta)$.

Instead of using the current parameter θ, the deep Q-network Mnih et al. (2015) computes the target action values based on an outdated parameter, i.e., the target network parameter θ^-, which keeps fixed over certain steps. The target value is then computed by

$$y = r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta^-). \tag{3}$$

Except for the use of the target Q-network with parameter θ^- , another important ingredient of the deep Q-network is the use of experience replay, where the agent's experience sample $e = (s_t, a_t, r_{t+1}, s_{t+1})$ of each step are stored in an experience replay memory dataset $\mathcal{D} = \{e_1, e_2, ...e_N\}$, and the training samples are randomly selected from this dataset to compute the loss of the Q-network.

The target Q-network reduces the correlation between action values and target values, while the experience replay disrupts the correlation of the observation samples sequence. By the combination of these two techniques, the deep Q-network can greatly improve the performance of the reinforcement learning algorithms Mnih et al. (2015); Van Hasselt et al. (2016).

3. DISTRIBUTED REINFORCEMENT LEARNING FRAMEWORK

3.1 Consensus algorithm

There have been many parallel algorithms designed for agents connected in a centralized communication graph. For example, the master-slave graph Li et al. (2014), where a server (master) is connected with several working agents (slave). Although this communication structure is simple and effective, the massive communication burden on the master is a prime shortcoming of this graph since that all working agents are required to communicate with the server at every step Suresh et al. (2017). This issue may be especially serious if the communication system is with high latency or low bandwidth. To circumvent the possible communication jam on the master, the decentralized communication graph is designed, where no center is required and each agent only shares its information with neighbors.

Assuming that the decentralized communication topology is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Specifically, $\mathcal{V} = \{1, 2, ...N\}$ and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ denotes the sets of agents and edges, where the communication between any two agents i and j is only allowed if the edge $(i, j) \in \mathcal{E}$. The $N \times N$ connectivity matrix $W = [w_{ij}]$ is further formalized to describe the connection strength of the N agents over the graph, where the value of w_{ij} means the degree of influence of agent i and agent j. Besides, $w_{ij} > 0$ only when $(i, j) \in \mathcal{E}$ or $i = j$, and $w_{ij} = 0$ represent the disconnection of agent i and j Olfati-Saber and Murray (2004).

The undirected topology is concerned in this paper, and the weighted connectivity matrix W of which should satisfy some conditions to ensure the consensus of all agents. That is, (i) $w_{ij} \in [0, 1)$, $\forall(i, j)$, (ii) $w_{ij} = w_{ji}$, $\forall(i, j)$, (iii) $\sum_{j=1}^{N} w_{ij} = 1$, $\forall i$.

Supposing that the model of agent i is denoted by a row vector θ_i, the update rule of consensus algorithm for this agent is described by

$$\theta_i' = \sum_{j=1}^{N} w_{ij}\theta_j. \tag{4}$$

where θ_i' denotes the updated model parameter of θ_i with a single consensus communication.

The update for the model parameters of all agents using the consensus algorithm is then defined as

$$\boldsymbol{\theta}' = \mathcal{C}(\boldsymbol{\theta}, W) = W\boldsymbol{\theta}. \tag{5}$$

where the matrices $\boldsymbol{\theta} = [\theta_1, \theta_2...\theta_N]^T$ and $\boldsymbol{\theta}'[\theta_1', \theta_2'...\theta_N']^T$ represent all agents' model parameters before and after the consensus update, respectively. θ_i is the ith row of $\boldsymbol{\theta}$, with N being the number of agents in the graph, .

The consensus algorithm makes all agents approach the mean value $\overline{\theta} = \frac{1}{N}\sum_{i=1}^{N} \theta_i$ only with the communication among connected agents by repetitively calculating the local average of the neighbors using (5).

3.2 Distributed deep Q-networks

Some researchers have proposed the asynchronous training framework for multi-agent reinforcement learning algorithms Mnih et al. (2016); Ong et al. (2015), where each agent asynchronously updates the Q-network on the central agent based on the experience samples on interacting with the environment. In this paper, we consider the scenario that multiple agents are connected in a communication topology without a central node, and each agent is only allowed to communicate with its neighbors. In addition, any agent's experience samples cannot be shared among agents due to the limitation of communication bandwidth and the privacy-preserving issues.

We now propose the distributed reinforcement learning framework for platoon vehicles, the aim of which is to train the Q-networks on all vehicles' combined experience without the sharing of experience data samples. First, the Q-networks of all vehicles are initialized, and each vehicle interacts with the environment (its front and back vehicles) to accumulate the experience samples in its replay memory. When the replay memory is filled with experience samples, each vehicle starts to train its Q-network, and the replay memory would be updated by replacing the old samples with the new samples. Unlike the single-vehicle situation, the training process of each vehicle in a platoon has been modified and changed to a two-phase procedure. In the first phase, each vehicle trains its Q-network based on its own replay memory, while these vehicles communicate with their connected neighbors in the second phase to globally update their Q-networks with the consensus communication.

By this means, each vehicle can learn other vehicles' experience without any of the vehicles having to reveal their experience data samples to other vehicles. Moreover, By running different vehicles in parallel, each vehicle can

learn other vehicles' experience samples simultaneously, which also reduces the correlation of experience samples generated by a single vehicle and improves its performance. For the multi-vehicle platoon scenario, all vehicles are initialized with the same Q-network and have their own replay memory. The pseudo-code for the distributed training method for deep Q-networks is summarized in Table 1.

Table 1. The process of distributed deep Q-networks

Algorithm 1: Distributed training for deep Q-networks

Inputs: The structure of the Q-network, the number of vehicles N, and the weighted connectivity matrix W of the communication topology.
Outputs: The optimal Q-network parameter θ^*.
1: Initialize the Q-network parameter θ_i and the corresponding target network parameter θ_i^- of each vehicle i.
2: Each vehicle i interacts with its front and back vehicles and records the experience sample ${}^ie = (s_t^i, a_t^i, r_{t+1}^i, s_{t+1}^i)$ in its replay memory dataset ${}^i\mathcal{D}$.
3: Train the Q-network parameter θ_i based on the randomly selected samples from the replay memory dataset ${}^i\mathcal{D}$, and update its corresponding target network parameter θ_i^- after each T steps.
4: Globally update the Q-network parameter θ_i of all vehicles over the graph using the consensus algorithm.
5: Back to step 2 with the updated θ_i.
6: Check the termination criterion (such as a given number of iterations).
7: Return the optimal Q-network parameter θ^*.

4. SIMULATION AND DISCUSSION

4.1 Platoon control using distributed Q-networks

One of the most important parts for reinforcement learning is the design of reward based on the current state and action. Here, we first consider the single vehicle scenario, where its environment consists of its front and back vehicles. The vehicle's current state contains its current position, velocity, and the positions of its front and back vehicles, that is

$$ {}^is_t = ({}^ip_t, {}^iv_t, {}^{i+1}p_t, {}^{i-1}p_t,). \tag{6}$$

where ${}^is_t$ is the current state of the ith vehicle, ${}^ip_t$ and ${}^iv_t$ are the position and velocity of the ith vehicle at tth step, respectively.

The reward is defined as follows, which consists of three terms. The first term of ${}^ir_{t+1}$ is to punish the distance between the ith vehicle position ${}^ip_{t+1}$ and the midpoint of the front and back vehicles, the second and third terms are used to punish the gap between the current velocity ${}^iv_{t+1}$ and v_0 and the accelerated velocity, respectively.

$$\begin{aligned} {}^ir_{t+1} = &- |{}^{i+1}p_{t+1}/2 + {}^{i-1}p_{t+1}/2 - {}^ip_{t+1}| \\ &- \alpha_1|{}^iv_{t+1} - v_0| - \alpha_2|a_t|, \end{aligned} \tag{7}$$

where ${}^ir_{t+1}$ is the corresponding reward from the environment based on action a_t in state ${}^is_t$, v_0 is the user-defined constant velocity of the leader vehicle. α_1 and α_2 are two tunable parameters to balance different rewards.

The task of the single vehicle is to learn how to keep the same separation distance to its front and back vehicles while moving forward, based on the reward calculated by (7) from the environment. Also, the vehicle would receive another positive reward if it fulfills the task or gets a negative punishment if it collides with its neighbors.

For this multi-vehicle platooning problem, there are two typical communication topologies as shown in Figure 1, where Figure 1(a) and Figure 1(b) can be taken as a master-slave and a decentralized graph, respectively. As discussed in section 3.1, the decentralized graph is more suitable for real-life practice.

Besides that no central node and no sharing of experience samples among vehicles are required for this distributed training framework, another superiority is the flexibility and expansibility of the decentralized communication topology. That is, the maximum communication burden on each vehicle will not rise with the increase in the number of vehicles over the platoon, and the effectiveness of this platoon is unaffected in switching communication topology Jiang et al. (2012) and the situation that any new vehicle participate in or leaves this platoon, on the condition that the communication platoon always has a spanning tree.

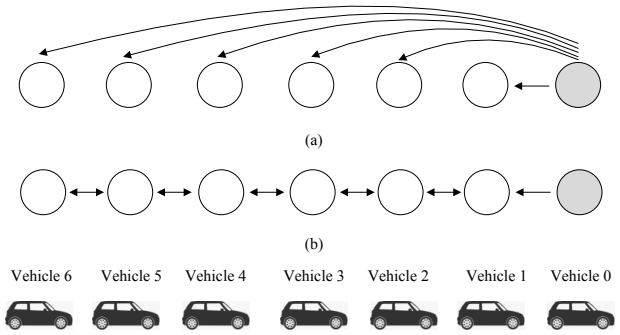

Fig. 1. Two typical communication topologies for platooning.

The proposed distributed training algorithm is suitable for any decentralized communication topology with a spanning tree. In this platooning simulation study, we take the topology in Figure 1(b) as an example, where the leader vehicle (indexed by 0) and six follower vehicles (indexed from 1 to 6) are connected over the platoon. In this platoon, the leader vehicle is moving at a constant speed, and each follower vehicle can only obtain its front and back vehicles' position. Furthermore, local communication on Q-networks is only required among directly connected neighbors. It is worth noting that we suppose there is a virtual back vehicle for vehicle 6, the position of which equals the position of vehicle 0 minus 7 times the given space between two neighboring vehicles. The whole simulation study is implemented in Tensorflow using python.

4.2 Simulation results

After initialized randomly from a normal distribution, the Q-network of each vehicle is locally optimized based on its own empirical risk, and then the consensus algorithm is

used to globally update all vehicles' Q-networks. As the actual Q value changes dramatically over iterations, we use the moving Q value $Q(t)$ to evaluate the performance of each vehicle in this simulation, which is calculated by the following equation

$$Q(t) = \begin{cases} \hat{q}(t) & t = 1 \\ 0.99Q(t-1) + 0.01\hat{q}(t) & t > 1, \end{cases} \tag{8}$$

where $\hat{q}(t) = Q(s_t, a_t, \theta)$ is the value of action based on the current Q-network.

Figure 2 shows the decrease in Q-network loss of all the vehicles, where each point is the mean value of 100 iterations. All vehicles take random actions to accumulate their experience replay memory and do not update their Q-networks in the first 3000 iterations, the loss of which is thus omitted in the figure. We can find from the partially magnified figure that all vehicles' loss decrease dramatically and converge to each other in the first 200 training iterations, and then converge to the optimal model in union.

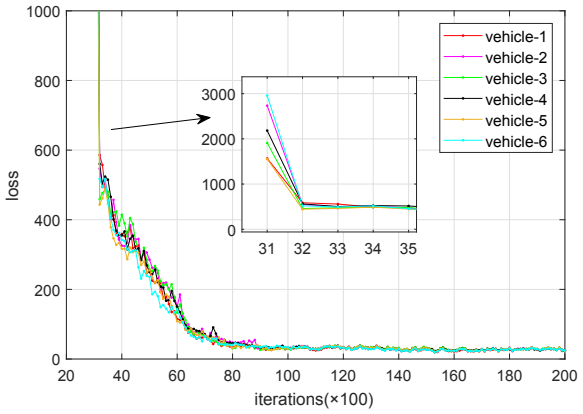

Fig. 2. The loss of each vehicle.

Figure 3 details the curves of all vehicles' moving Q value. In the first 3000 iterations, all vehicles interact with their front and back vehicles with random action to accumulate their experience data samples. In this stage, all the vehicles do not update their Q-networks, and thus the moving Q values randomly fluctuate around their initial values, which are based on their initial parameters of Q-networks. After 3000 iterations, all the vehicles start to learn and update their Q-networks, and their moving Q values converge to each other in a few hundred iterations and decrease dramatically in union. In this stage, all vehicles have not learned how to reach the midpoint of their front and back vehicles, and thus receive negative rewards. The moving Q values stop dropping and start to increase around 4500 iterations. After that, these vehicles learn how to form a platoon and the corresponding moving Q values begin to increase and reach convergence.

Figure 4 describes the performance of all vehicles' relative positions in the last episode. The positions of all vehicles are first randomly initialized in order, and they learn to form the required platoon while updating their Q-networks. After 3000 iterations, all vehicles form the platoon and move forward with the same separation distance between the front and back vehicles, and there is no vehicle collision during this process. It is clear from

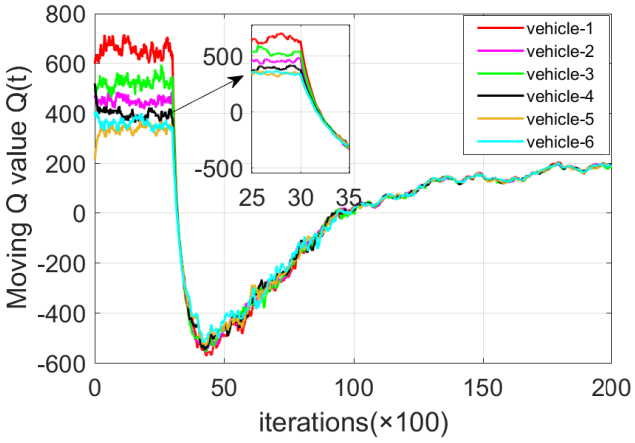

Fig. 3. The moving Q value of each vehicle.

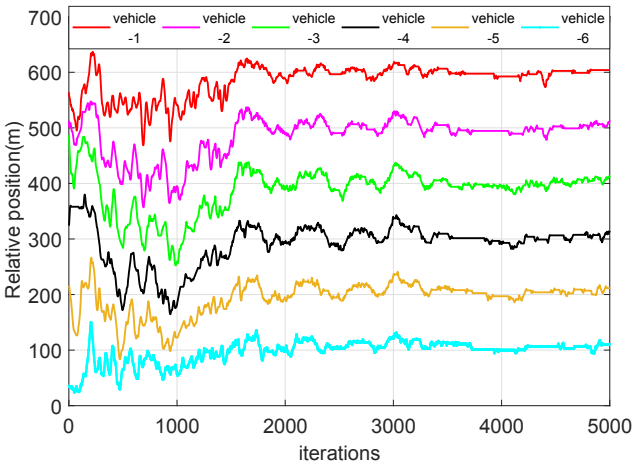

Fig. 4. The relative position of each vehicle.

the figure that there are still some fluctuations after the vehicles form a stable platoon, and the reasons may be multifaceted for this phenomenon.

First, there is a small possibility for each vehicle to choose an action randomly other than taking the best action because of the exploration strategy, which is a very important part of reinforcement learning. Second, DQN is designed for those problems with discrete and limited actions, while the actions of this platooning control problem are original continuous and unlimited, which makes DQN not directly suitable. In this simulation, we transfer the continuous actions into limited actions using discretization with equal space, which makes it impossible for the vehicles to always choose the best action. Third, the environment for each vehicle, i.e. its front and back vehicles, is varying, and all vehicles are with strong coupling, which means that the destabilization of any vehicle will affect all the vehicles in this platoon. Although the simulation results are with some fluctuations, vehicles can form the required platoon and move at roughly equal space, which is suitable for practice and more in line with the actual human driving situation. In practical application, the dynamics of the vehicles are not required, and there is no limit on the number of vehicles over an arbitrary communication topology with a spanning tree. Each vehicle only needs to obtain its front and back vehicles' positions to learn how to control the accelerator for this platooning problem.

5. CONCLUSION AND FUTURE WORK

In this paper, we propose a distributed training framework for deep Q-networks to deal with the platoon control problem using a reinforcement learning method. Multiple follower vehicles and a leader vehicle are connected over a decentralized platoon. This multi-vehicle platooning problem is decomposed into many small single vehicle tasks, where each vehicle takes its front and back vehicles as the reinforcement learning environment and learns to keep the same space to them. All the vehicles only need to communicate and obtain the positions of its front and back vehicles in this platoon and the sharing of experience samples is not allowed because of privacy concerns and communication bandwidth limitation, which is more suitable for the practical situation. After the optimization of every vehicles' Q-networks based on their own experience samples, the consensus algorithm is designed to allow all vehicles to converge to each other. In this way, we change the learning process of Q-network into a two-phase update process, where the Q-network of each vehicle is locally optimized based on its own experience first, and the Q-networks of all vehicles are then globally updated using the consensus algorithm.

At last, the simulation study shows that the Q-networks of all vehicles reach consensus in 200 training iterations and converge to the optimal model around 4500 iterations. After randomly initialized in the last learning episode, all the vehicles learn to form the required platoon in 3000 iterations and move forward at roughly equally separation distance.

REFERENCES

Gurney, K. (2014). *An introduction to neural networks*. CRC press.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. (2018). Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*.

Jiang, H., Bi, Q., and Zheng, S. (2012). Impulsive consensus in directed networks of identical nonlinear oscillators with switching topologies. *Communications in Nonlinear Science and Numerical Simulation*, 17(1), 378–387.

Knorn, S., Chen, Z., and Middleton, R.H. (2015). Overview: Collective control of multiagent systems. *IEEE Transactions on Control of Network Systems*, 3(4), 334–347.

Li, M., Andersen, D.G., Park, J.W., Smola, A.J., Ahmed, A., Josifovski, V., Long, J., Shekita, E.J., and Su, B.Y. (2014). Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, 583–598.

Li, S.E., Zheng, Y., Li, K., and Wang, J. (2015). An overview of vehicular platoon control under the four-component framework. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, 286–291. IEEE.

Liang, Z., Chen, J., and Wang, Y. (2019). Equivalent acceleration imitation for single wheel of manned lunar rover by varying torque on earth. *IEEE/ASME Transactions on Mechatronics*, 1–1. doi:10.1109/TMECH.2019.2953330.

Lu, N., Cheng, N., Zhang, N., Shen, X., and Mark, J.W. (2014). Connected vehicles: Solutions and challenges. *IEEE internet of things journal*, 1(4), 289–299.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529.

Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., et al. (2015). Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*.

Olfati-Saber, R. and Murray, R.M. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Automat. Contr.*, 49(9), 1520–1533.

Ong, H.Y., Chavez, K., and Hong, A. (2015). Distributed deep q-learning. *arXiv preprint arXiv:1508.04186*.

Ploeg, J., Shukla, D.P., van de Wouw, N., and Nijmeijer, H. (2013). Controller synthesis for string stability of vehicle platoons. *IEEE Transactions on Intelligent Transportation Systems*, 15(2), 854–865.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Suresh, A.T., Yu, F.X., Kumar, S., and McMahan, H.B. (2017). Distributed mean estimation with limited communication. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3329–3337. JMLR. org.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Yang, S., Liu, Q., and Wang, J. (2016). A multi-agent system with a proportional-integral protocol for distributed constrained optimization. *IEEE Transactions on Automatic Control*, 62(7), 3461–3467.

Zhang, H., Zhang, Q., Liu, J., and Guo, H. (2018). Fault detection and repairing for intelligent connected vehicles based on dynamic bayesian network model. *IEEE Internet of Things Journal*, 5(4), 2431–2440.

Zhang, L. and Orosz, G. (2016). Motif-based design for connected vehicle systems in presence of heterogeneous connectivity structures and time delays. *IEEE Transactions on Intelligent Transportation Systems*, 17(6), 1638–1651.

Zheng, Y., Li, S.E., Li, K., Borrelli, F., and Hedrick, J.K. (2016). Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies. *IEEE Transactions on Control Systems Technology*, 25(3), 899–910.

Zheng, Y., Li, S.E., Li, K., and Ren, W. (2017). Platooning of connected vehicles with undirected topologies: Robustness analysis and distributed h-infinity controller synthesis. *IEEE Transactions on Intelligent Transportation Systems*, 19(5), 1353–1364.