Multi-agent Traffic Signal Control via Distributed RL with Spatial and Temporal Feature Extraction

Yifeng Zhang, Mehul Damani, and Guillaume Sartoretti(✉)

Department of Mechanical Engineering, National University of Singapore, Singapore, Singapore
e0576068@u.nus.edu , mpegas@nus.edu.sg
http://marmotlab.org

1 Introduction

The aim of traffic signal control (TSC) is to optimize vehicle traffic in urban road networks, via the control of traffic lights at intersections. Efficient traffic signal control can significantly reduce the detrimental impacts of traffic congestion, such as environmental pollution, passenger frustration and economic losses due to wasted time (e.g., surrounding delivery or emergency vehicles). At present, fixed-time controllers, which use offline data to fix the duration of traffic signal phases, remain the most widespread. However, urban traffic exhibits complex spatio-temporal patterns, such as peak congestion during the start and end of a workday. Fixed-time controllers [8,10] are unable to account for such dynamic patterns and as a result, there has been a recent push for adaptive TSC methods.

Reinforcement learning (RL) is one such adaptive and versatile data-driven method which has shown great promise in general robotics control. Recent works that have applied reinforcement learning to the traffic signal problem have shown great promise in alleviating congestion in a single traffic intersection [5–7,9,11]. A traffic network is composed of multiple such intersections and a network optimization problem can be broadly formulated as a centralized (single-agent) or decentralized (multi-agent) RL problem. In the centralized RL formulation, a global agent controls the entire traffic network and tries to minimize a global objective such as average trip time. However, centralized solutions for adaptive TSC are infeasible in practice due to the exponentially growing joint action and state space and the high latency associated with information centralization. To avoid the curse of dimensionality , decentralized approaches frame traffic signal control as a multi-agent RL (MARL) problem, where each agent controls a single intersection, based on locally-sensed real-time traffic conditions and communication with neighboring intersections [1,14]. In this work, we propose a framework for fully decentralized multi-agent TSC (MATSC) based on distributed reinforcement learning with parameter sharing [4], for improved scalability and

Supported by Singapore Technologies Engineering Ltd, under work package 3 of the "Urban Traffic Flow Smoothening Models" NUS-STE joint laboratory.

F. S. Melo and F. Fang (Eds.): AAMAS 2022 Workshops, LNAI 13441, pp. 106–113, 2022.
https://doi.org/10.1007/978-3-031-20179-0_7

performance. We design a spatial and temporal neural network, by relying on an attention mechanism and a recurrent unit, to extract spatial and temporal features about local traffic conditions at each intersection. We compare our framework with state-of-the-art MATSC methods in simulation, and show that our approach results in decreased average queue lengths and trip times, as well as increased average vehicle speeds and trip completion rates, both overall and during peak periods.

2 Method

2.1 Problem Formulation

We use a decentralized MARL formulation for MATSC. Each traffic intersection is controlled by a RL agent which only has access to local traffic conditions i.e., the agents have partial observability.

More formally, we consider the multi-agent extension of an MDP, which is characterized by a set of states, S, action sets for each of N agents, $A_1, ..., A_N$, a state transition function, $P : S \times A_1 \times ... \times A_N \rightarrow S'$, which defines the probability distribution over possible next states, given the current state and actions for each agent, and a reward function for each agent that also depends on the global state and actions of all agents, $R_i : S \times A_1 \times ... \times A_N \rightarrow R$.

We consider a partially observable variant in which an agent, i, can observe part of the system state $s_i \in \mathbf{S}$ as its observation $o_i \in \mathbf{O}$. The *state* of junction i at time step t comprises two vectors: The first one is a one-hot vector representing the current traffic phase, and the second indicates the number of vehicles on each incoming lane. In line with recent works [2], the observation space of each agent is composed of the state of its assigned junction, as well as the state of all directly connected neighboring intersections.

The action space $\mathbf{A}$ is defined as a set of non-conflicting phases. Specifically, at time step t, agent i will choose an action a_i^t from its own action space A_i as a decision for the next Δt_g period of time i.e., the intersection will be in the chosen phase from time step t to time step $t + \Delta t_g$. After the fixed duration of a given phase has elapsed, the agent may choose to continue with the same phase or choose a different phase and incur a transitional yellow phase penalty of duration Δt_y. In this work, we set Δt_g and Δt_y to $5s$ and $2s$ respectively.

Following [2], we define the reward structure as a short term metric which is calculated as the sum of the number of halting vehicles on the lane-area detectors.

2.2 Spatial and Temporal Perception Network

Given the dynamics of a traffic network, an agent (junction) needs to have both spatial and temporal awareness to make informed decisions. To enable this, we propose a network that comprises two units, a message aggregation unit for spatial feature extraction, and an RNN-based memory unit for temporal awareness. The detailed network structure is shown in Fig. 1. The attention-based message aggregation unit [13] allows the agent to learn to assign higher weights to

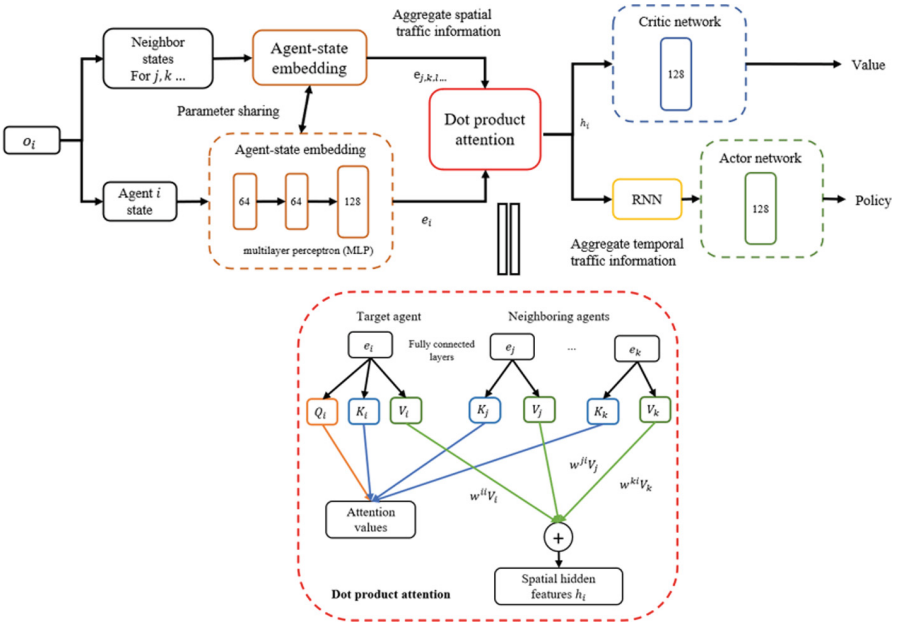

Fig. 1. Structure of the spatial-temporal perception neural network used in this work.

essential parts of the observations, i.e., concentrate more on the traffic states of neighboring intersections that might cause significant impacts on itself in future steps, while the recurrent unit allows it to utilize historical information to inform its current decision-making.

Specifically, we first map all the raw observations to a higher dimensional feature space i.e., mapping from low dimensional o_i to high dimensional e_i through multiple fully connected layers. Then, we obtain the query, key, value vectors of the attention mechanism (with same dimensions), by using three different sets of learned weights: $q_i = W_i \cdot e_i, k_i = W_k \cdot e_i, v_i = W_v \cdot e_i$. In our work, the parameters of the key, query, and value layers are shared among agents.

Next, we calculate the compatibility u_{ij} between the query q_i and k_j based on scaled dot production mechanism: $u_{ij} = (q_i^T \cdot k_j)/\sqrt{d}$, where d is the dimension of the vectors used for normalization. The attention weights for each query-key pair can be computed by: $\alpha_{ij}^h = softmax\,(u_{ij})$.

Finally, we calculate the output vector as the weighted sum of all the value vectors, using these learned attention weights: $h_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \cdot v_j$.

Second, we rely on a recurrent neural network (here, a Gated Recurrent Unit, GRU) [3] to extract temporal features from the agents' observations. Overall, through the proposed network structure, we are able to extract features in both the spatial (using attention) and temporal (using GRU) dimensions.

2.3 Learning Framework

We use the popular PPO algorithm for training the policy [12]. PPO's update rule prevents large changes to the policy, which is particularly desirable in our distributed, parameter-sharing setting where there is significant noise in computed gradients. We use the Adam Optimizer with learning rate $5e{-}5$, an episode length of 720 and a discount factor(γ) of 0.95.

Inspired by some of our previous works [4], we developed a hierarchical distributed learning framework to make use of parallelization and parameter sharing. Instead of learning a separate policy for each intersection, we use parameter sharing between intersections to learn a single, universal policy common to all agents (junctions) in the network. Our distributed framework instantiates multiple low-level "workers" (meta-agents) and a high-level coordinator called the "driver". Each worker is regarded as a multi-agent system and works in an identical but independent environment. The goal of the worker is to collect the experience of all learning agents in an environment. The driver uses the shared experience of all workers to update a global shared network at the end of each episode.

In addition to significant gains in wall-clock training time, our distributed learning framework has two main advantages. First, parameter sharing between agents reduces the instability of distributed MARL associated with independent learning by preventing drastic updates to the global network and thus ensuring that the environment is relatively stable from any single agent's perspective. In addition, the (shared) network weights are updated at the end of each episode to improve the individual rewards of each agents, implicitly leading to a common policy that aims at optimizing their common decisions, thus encouraging the formation of cooperative behaviors.

Second, the structure of the distributed framework is modular and can easily be adapted to run multiple state-of-the-art RL algorithms such as A3C and SAC.

Third, we note that our learning framework aims at offline (centralized) training before online (decentralized) execution. That is, our learning agents will first be trained in simulation. Then, the trained policy can be frozen and deployed in the real world in a fully decentralized manner, i.e., based on local sensing and communication among neighboring agents. The advantages of this offline, centralized training, decentralized execution design choice are:

1. Offline training is cheap, since we do not need to be concerned about the potential threats (e.g., congestion, accidents) caused by unreasonable behaviors that would result from agents freely exploring their state-action space during early training.
2. Offline training can still allow a sim-to-real solution with high portability: the policy can be trained in simulation by relying on real-world data, if available; alternatively, the trained policy may be fine-tuned under real-world traffic conditions after deployment, to truly reach a near-optimal controller.

3 Experiments and Discussion

We conducted our simulation experiments using a Manhattan traffic network based on the benchmark method MA2C, and the same exact traffic demand [2].

As illustrated in Fig. 2, there are a total of 25 homogeneous signalized intersections in this grid network, where each one is formed by two-laned, horizontal arterial streets with a speed limit of 20 m/s, and one-laned, vertical avenues with a speed limit of 11 m/s. Each intersection contains five permissible phases: East-West straight phase, East-West left-turn phase, and three straight and left-turn phases for East, West, and North-South, respectively. The lane-area detectors are installed near the stop line of the intersection, with a length of 50 m, which are shown as the blue rectangles in Fig. 2.

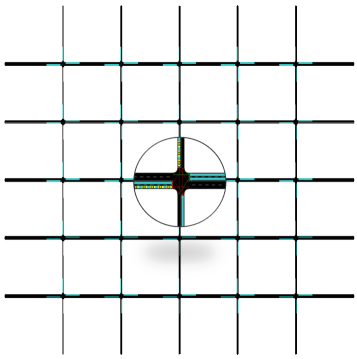

Fig. 2. Simulated grid traffic scenario with 25 homogeneous intersections, adapted from the benchmarks used in [2].

We compare our method (dDRL-Att) with a conventional greedy controller (one-step optimal controller with respect to the same metric used by our dDRL approach) and three learning based methods (MA2C, IA2C, and IQL-LR) [2]. These last three baselines are decentralized multi-agent RL algorithms, where each agents learns an independent policy depending on its local traffic conditions. IQL-LR (linear regression based independent Q-learning) is a fully scalable Q-learning algorithm, where each local agent learns a local Q network based on its own action, while regarding other agents as part of the environment. IA2C extends the idea of IQL to the advantage actor-critic (A2C) algorithm, where the actor network directly learns a policy that maps the input state to a probability distribution over actions, while the critic network learns to predict the state value as a baseline for calculating the advantage value guiding the actor's training. However, IQL-LR and IA2C suffer from non-stationarity, which mainly stems from partial observability and limited communication among agents. Multi-agent A2C (MA2C) attempts to mitigate this common problem of independent learning algorithms by proposing two improvements. Firstly, MA2C includes the states and the sampled latest policies (fingerprints) of neighboring agents in the local state of each agent. Second, it introduces a spatial discount factor to decrease the impacts of both the observations and rewards of neighboring agents, thus encouraging agents to concentrate more on their local traffic conditions. In some ways, we note that the attention weights learned by our approach play a similar role as this spatial factor, but with the added advantage that our weights are dynamic, and thus can adapt to the different states of neighboring agents.

We also include a non-attention version of our method (dDRL) for a simple ablation study on this aspect. The comparison test results are shown in Fig. 3,

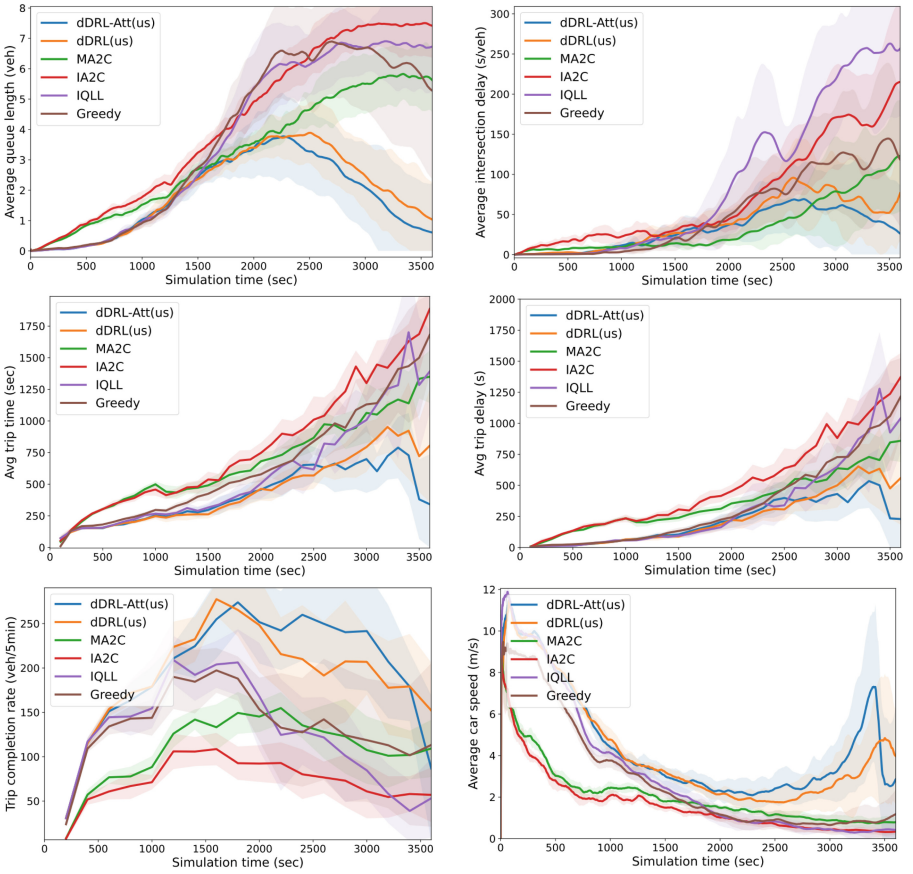

Fig. 3. Evaluation results for 10 episodes in the considered 5×5 Manhattan network. The solid lines show average values, while the standard deviations are shaded.

where we measure and record the traffic metrics at each simulation step and then calculate the averages and deviations over a fixed set of 10 test episodes (i.e., same traffic conditions for all algorithms for fair comparison).

From these evaluation results, we first observe that the queue length and delay time curves of both IQL-LR and IA2C show a monotonically increasing trend. This trend indicates that as the simulation time increases, the vehicles gradually start accumulating on the incoming lanes, leading to growing congestion. Compared to IQL-LR and IA2C, the queue length and delay time curves for MA2C rise significantly slower, indicating that MA2C is able to learn an effective policy. However, it still lacks the ability to reduce congestion in saturated networks, especially when the traffic gets heavier after 2000s.s. Finally, our distributed deep RL methods (dDRL-Att and dDRL) outperform the above-discussed baselines on handling peak-time traffic as well as recovering from saturated and congested traffic networks. This is evident from the queue length

Table 1. Temporal averages and peaks of all algorithms averaged over 10 episodes in the considered 5×5 Manhattan traffic network (best value on each line in bold).

Metrics (Average)	Temporal Averages					Temporal Peaks				
	dDRL-Att (us)	dDRL (us)	MA2C	IA2C	IQL-LR	dDRL-Att (us)	dDRL (us)	MA2C	IA2C	IQL-LR
Queue length (veh)	**1.800**	1.974	3.204	4.214	3.762	**3.816**	3.939	5.944	7.555	6.942
Speed (m/s)	**4.468**	4.223	2.003	1.571	3.041	13.361	**14.176**	11.656	11.513	13.978
Intersection delay (sec)	**30.984**	38.837	34.985	72.959	92.591	**70.929**	98.612	124.005	216.112	266.912
Trip completion rate (veh/s)	**0.990**	0.955	0.546	0.366	0.634	**2.300**	**2.300**	1.700	1.400	2.100
Trip time (sec)	**447.519**	460.735	727.387	788.501	462.862	**2330.000**	2745.000	2938.000	3445.000	2373.000

curve for dDRL-Att and dDRL, which rapidly trends downwards after 2400 s,s, showing that the policy learned by our method is more sustainable and stable. In particular, compared to dDRL, dDRL-Att shows better performance in detecting and handling high-volume traffic, as evidenced by the fact that our approaches are able to handle peaks of traffic and return to baseline conditions faster (see Fig. 3, top left). We also observe that dDRL outperforms MA2C and other methods on completion rate, highlighting its effectiveness at maximizing throughput while minimizing congestion. The same conclusion can be obtained from the average speed metric, with higher values signalling fewer halts and higher overall vehicle flow.

Second, Table 1 summarizes the quantitative results for comparing different methods. From the table, we observe that our proposed dDRL-Att method performs better than all baselines in all the temporal average metrics. Finally, compared to dDRL, our multi-head-attention-based spatial and temporal network performs better, which showcases its effectiveness at extracting features that allow for more informed decision making.

4 Conclusion and Future Work

In this work, we introduced a framework for fully decentralized multi-agent TSC (MATSC) based on distributed reinforcement learning with parameter sharing, which relies on an attention mechanism and a recurrent unit for the extraction of spatial and temporal features. Through experimental results, we demonstrated that our proposed framework results in decreased average queue lengths, and increased average vehicle speeds and trip completion rates. Future work will focus on developing and improving techniques such as credit assignment that allow for increased cooperative behavior between agents in an effort to achieve better performance on global long-term metrics while staying in the regime of partial observability and local sensing.

Acknowledgements. Authors would like to thank Dr. Teng Teck Hou for his feedback on earlier drafts of this manuscript as well as for research discussions during the elaboration of this work.

References

1. Camponogara, E., Kraus, W.: Distributed learning agents in urban traffic control. In: Pires, F.M., Abreu, S. (eds.) EPIA 2003. LNCS (LNAI), vol. 2902, pp. 324–335. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-24580-3_38
2. Chu, T., Wang, J., Codecà, L., Li, Z.: Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE Trans. Intell. Transp. Syst. **21**(3), 1086–1095 (2019)
3. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
4. Damani, M., Luo, Z., Wenzel, E., Sartoretti, G.: PRIMAL$_2$: pathfinding via Reinforcement and Imitation Multi-Agent Learning - Lifelong. IEEE Robot. Autom. Lett. **6**(2), 2666–2673 (2021). https://doi.org/10.1109/LRA.2021.3062803
5. Gao, J., Shen, Y., Liu, J., Ito, M., Shiratori, N.: Adaptive traffic signal control: deep reinforcement learning algorithm with experience replay and target network. arXiv preprint arXiv:1705.02755 (2017)
6. Garg, D., Chli, M., Vogiatzis, G.: Deep reinforcement learning for autonomous traffic light control. In: 2018 3rd IEEE International Conference on Intelligent Transportation Engineering (ICITE), pp. 214–218. IEEE (2018)
7. Genders, W., Razavi, S.: Evaluating reinforcement learning state representations for adaptive traffic signal control. Procedia Comput. Sci. **130**, 26–33 (2018)
8. Hunt, P., Robertson, D., Bretherton, R., Winton, R.: Scoot-a traffic responsive method of coordinating signals. Technical report (1981)
9. Li, L., Lv, Y., Wang, F.Y.: Traffic signal timing via deep reinforcement learning. IEEE/CAA J. Automatica Sinica **3**(3), 247–254 (2016)
10. Luk, J.: Two traffic-responsive area traffic control methods: scat and scoot. Traffic Eng. Control **25**(1) (1984)
11. Mousavi, S.S., Schukat, M., Howley, E.: Traffic light control using deep policy-gradient and value-function-based reinforcement learning. IET Intel. Transp. Syst. **11**(7), 417–423 (2017)
12. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
13. Vaswani, A., et al.: Attention is all you need. Adv. Neural Inf. Process. Syst. **30** (2017)
14. Wei, H., et al.: Presslight: learning max pressure control to coordinate traffic signals in arterial network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1290–1298 (2019)