MULTI-AGENT REINFORCEMENT LEARNING FOR OPTIMIZING TRAFFIC SIGNAL TIMING

Areej Salaymeh[1], Loren Schwiebert[1] and Stephen Remias[2]

[1]Department of Computer Science, Wayne State University, Detroit, USA

[2]Civil and Environmental Engineering, Wayne State University, Detroit, USA

ABSTRACT

Designing efficient transportation systems is crucial to save time and money for drivers and for the economy as whole. One of the most important components of traffic systems are traffic signals. Currently, most traffic signal systems are configured using fixed timing plans, which are based on limited vehicle count data. Past research has introduced and designed intelligent traffic signals; however, machine learning and deep learning have only recently been used in systems that aim to optimize the timing of traffic signals in order to reduce travel time. A very promising field in Artificial Intelligence is Reinforcement Learning. Reinforcement learning (RL) is a data driven method that has shown promising results in optimizing traffic signal timing plans to reduce traffic congestion. However, model-based and centralized methods are impractical here due to the high dimensional state-action space in complex urban traffic network.

In this paper, a model-free approach is used to optimize signal timing for complicated multiple four-phase signalized intersections. We propose a multi-agent deep reinforcement learning framework that aims to optimize traffic flow using data within traffic signal intersections and data coming from other intersections in a Multi-Agent Environment in what is called Multi-Agent Reinforcement Learning (MARL).

The proposed model consists of state-of-art techniques such as Double Deep Q-Network and Hindsight Experience Replay (HER). This research uses HER to allow our framework to quickly learn on sparse reward settings. We tested and evaluated our proposed model via a Simulation of Urban MObility simulation (SUMO). Our results show that the proposed method is effective in reducing congestion in both peak and off-peak times.

KEYWORDS

Multi-agent, Deep learning, Traffic signal timing, Reinforcement learning.

1. INTRODUCTION

Transportation is the cornerstone of modern economies, and when it comes to moving things around, there is nothing that is more important than the road transportation system. Every day, millions of cars and trucks move through our roads to transport people and goods, so having an efficient and reliable transportation system is key to our lives. However, as the number of vehicles increases, pressure on the transportation system also increases, thus leading to more congestion, which has negative economic and environmental consequences. One possible solution is to improve the efficiency of the current infrastructure by intelligently optimizing traffic signal timing plans.

One of the most common traffic control devices is the traffic signal. Traffic signal control policy can be a key factor in improving traffic, as it controls the flow of vehicles on roads. In practice, not all traffic signal times are optimized, and vehicles are often waiting for numerous timing cycles, causing delay. Having adjustable timings for signals can lead to less congestion. However, solving the traffic control problem has three key challenges, modelling, scalable data, and optimization [1].

David C. Wyld et al. (Eds): AIAP, SIGML, CNSA, NIAI - 2021
pp. 21-36, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.110102

In the past few decades, several adaptive traffic control systems were designed and applied. Examples include Split Cycle Offset Optimization Technique (SCOOT) [3], which is a real time system that uses sensors to collect data about traffic, then adjusts traffic signals accordingly in small steady increments to avoid major disruptions to or fluctuations in the traffic. Sydney Coordinated Adaptive Traffic System, or SCATS, is another intelligent transportation system used in Australia. PRODYN is an algorithm that uses Forward Dynamic Programming (FDP) at the intersection level to optimize traffic signal times [6]. Although these systems have been adopted and installed in different places around the world, they have some substantial deficiencies such as high infrastructure costs, intense computation, and limited scalability due to the lack of global information.

On the other hand, various adaptive control techniques have been proposed to the solve traffic signal control optimization problem such as fuzzy logic [7], genetic algorithms [8], and approximate dynamic programming [9]. These approaches are modeled using limited information, resulting in reduced accuracy and scalability.

Reinforcement learning (RL) has been shown to be a promising alternative to approximate an optimal decision-making policy for complex transportation systems [2,5]. In RL, the basic model is formed as a Markov Decision Process (MDP) which makes it the best fit for the traffic signal control problem [2,4]. The traffic signal control problem can be solved using sequential decision making and is a complex task as dynamic traffic flow changes can make the environment unpredictable. Traditional RL methods such as tabular methods and Linear Regression (LR) are difficult to apply here due to scalability and optimality challenges [16].

Figure 1 shows the basic RL framework. In RL, an agent is used to observe the environment either fully or partially and is trained to eventually behave in an optimal way in a certain environment state, which will lead to a solution. The agents interact in discrete time steps with the environment to find the optimal solution. The main goal of the agent is to select actions that will lead to maximizing the total future reward. The agent keeps improving its policy to find the optimal policy.

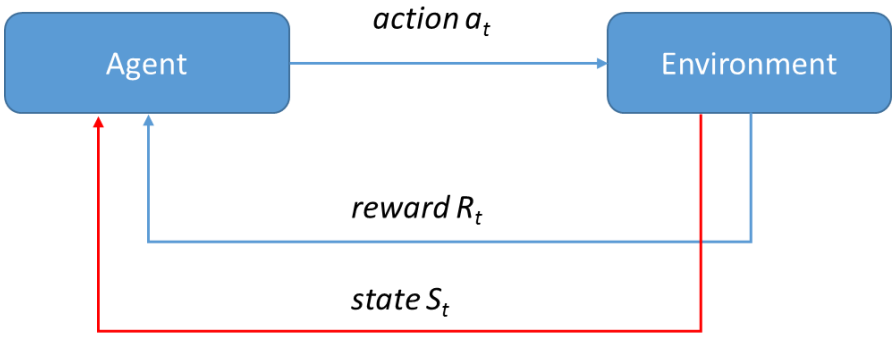

Figure 1: Reinforcement Learning Agent-Environment Cycle

Generally, for RL, algorithms can be model-free or model based [10,11]. In model-based RL, the MDP is estimated, where agents act in the world and observe state. The model-based RL will converge to the optimal policy. On the other hand model-free algorithms approximate the state-action function *Q(s,a)* instead of estimating the state-value function V(s). The model-free value-based policy seeks the optimal value function, to solve an MDP. It is possible to do this procedure by computing and updating the state-action value function iteratively, described by Q. Another model-based type is policy based RL, where the RL will learn from the optimal policy instead of the MDP or using Q functions [10].

In RL, a system can have multiple agents, such as in games, robotics, and traffic management systems. In particular, MARL [17] has multiple autonomous agents operating in a common environment, where each agent will attempt to optimize its goals through interactions with the other agents and the environment as a whole. MARL algorithms can essentially range from completely cooperative to fully competitive [19,20]. In the cooperative setting, the agents work together to maximize a shared long-term return. In competitive MARL algorithms, agents try to focus on maximizing their own goals. Agents can have a behavior between cooperative and competitive, such as cases when and agent can cooperate with some agents temporarily.

In recent years, deep learning has been introduced into RL to enhance optimization and scalability. DQN uses a deep network to approximate the Q value [10]. In deep learning, input samples are randomized to achieve a balanced input sampling across training batches. However, in RL, the input space is constantly changing as the model learns, which presents challenges for the Q-value approximation process. To address this, DQN uses experience replay and target networks to slow down changes. Experience replay uses a buffer to store historical state, action and rewards that will be used in training, thus giving us a way that will find Q. Furthermore, we have two networks to store Q values [10]. One is continuously updated, while the second, the goal network, is synchronized occasionally with the first network. We use the target network to extract the Q value, so that adjustments are more predictable for the target value [12].

Sparse reward settings are one of the obstacles for reinforcement learning [16]. That is, where the agent receives a reward only when meeting the target level. Most reinforcement learning algorithms, however, require input from a reward to learn how to solve the problem, or learn at all. Therefore, by not earning any reward, most algorithms are doomed to fail on the specific task if they never reach the target state. Special exploration techniques are required to achieve challenging target states and to eventually learn based on the reward obtained. Hindsight Experience Replay (HER) is one of those learning techniques that allows us to quickly learn on sparse reward settings [13]. HER's uniqueness is that it is a very simple and intuitive extension, which can be incorporated into many algorithms of off-policy learning. With this simple addition the algorithms are now able to solve sparse reward settings, which they would not be able to do without HER.

In this paper, the problem of traffic signal control is addressed by using a multi-agent deep reinforcement learning model that will use Double DQN [22] and HER to optimize traffic signal timing for multiple traffic signals simultaneously. We propose an intelligent multi-agent system for four-phase signalized intersections, in which the agents are cooperative and share vital information to achieve a stable model and scalable network. In the following sections we will go over related work, the proposed framework, the simulation, and the results.

2. RELATED WORK

The implementation of RL have been studied extensively to find the optimal traffic signal timing policy since the 1990s [1,5,16]. Earlier work was limited to Tabular Q-learning in RL. Some researchers proposed a discrete model-based using co-learning feature, where vehicles can vote and contribute in a traffic signal’s decision in order to reduce overall waiting time [5].

These methods can handle only discrete state representations. Here, the states are defined by discrete values such as location of vehicles, number of waiting vehicles or the waiting queue length. However, constructing the lookup table to store the state-action pair values requires massive storage space and is limited to only a small set of states. However, the complexity in real world road traffic needs high dimensional state and action spaces [18].

Recent work using deep reinforcement learning has shown impressive results in handling more complex state representations. It achieves higher performance than the exhaustive representation by a lookup table. The earliest attempt to deploy Deep Q-Networks (DQN) in the traffic signal control problem was by Li, et al.[23] Here, the researchers used deep stacked auto-encoders (SAE) to learn the Q-function in order to control an isolated intersection. Genders [24] developed a DQN to control a four-phase signalized intersection, but it made impractical assumptions by ignoring the fixed phasing sequence of the traffic signal. Mousavi [25] showed the superiority of deep policy-gradient over the value-function based approach in finding more stable control policies. All the previous methods had scalability issues and hence either control a single intersection or a small traffic network.

Despite all the improvements in scalability gained by using deep reinforcement learning in training a centralized RL, it is still impractical for large networks. One alternative approach is to apply multi-agent reinforcement learning, where agents are cooperative and share vital information [14,15].

Although there is a rich history of using RL, most of the work on the traffic control problem has concentrated exclusively on independent agents and only a few of them used MARL. Wiering [26] proposed discrete model-based tabular Q-learning while [12] used model-free Q-learning which both work only for a small state space. Abdoos extended his work [27] to have a holonic Q-learning multi-agents system using graph-based algorithm and divide the problem into sub-problems, dividing the fifty intersections to thirteen holons. However, this approach is impractical due to scalability issues as the need to model the network needs a large storage space and intense computation.

3. PROPOSED METHOD

The problem tackled in this work is defined as follow: Given the state (environment dynamics), the agents need to select the best action (traffic signal phase) in order to maximize the total reward, hence, improving traffic capacity and reducing congestion. In this section, we present a detailed description of the design of the framework.

3.1 Designing the Agent

An agent represents a phase rather than a single traffic signal, in order to not have conflicting phases that are physically impossible or could lead to crashes. During the simulation, the agent will acquire the state from the Simulation of Urban MObility (SUMO) [21] environment and use that as an input to calculate the reward function. SUMO is open source and widely adopted. Information about the previous simulation step data is drawn from the memory. Once the reward function is calculated, the agent will select a new action and update the environment. All these steps are represented in Figure 2.

In our work, we are relying on data that is generated by the simulation, which will enable us to test different scenarios and different traffic flows. In real-world situations, these data are gathered from vehicle detection, which are sensors that are installed in or above the roadway. These traffic detectors register data in real time including vehicle count, speed, and occupancy.

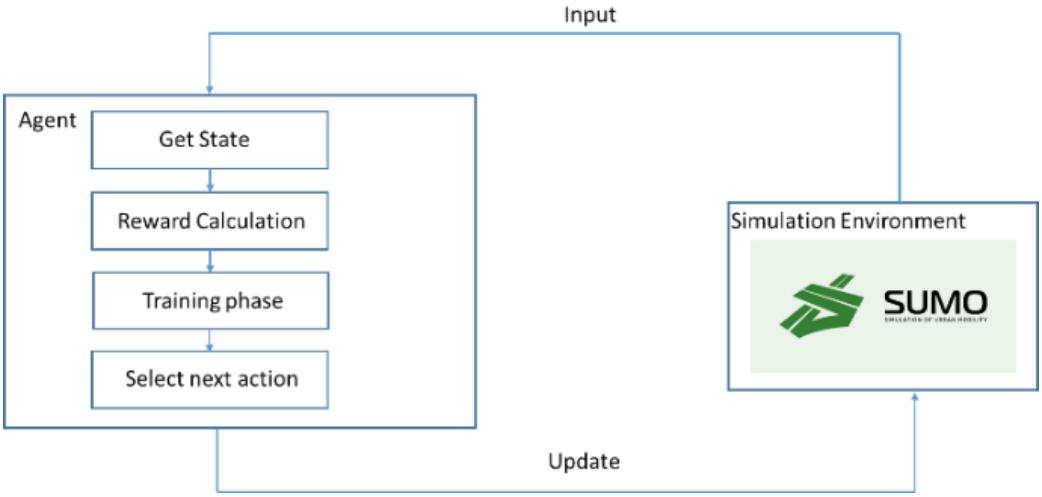

Figure 2: Workflow of our proposed method

3.2. Designing MA RL for the proposed method

A multi-agent system is a system where a number of agents share an environment and can communicate with each other. Agents can have a full view of the environment or a partial one. With a single agent, it is relatively easy to design a reward function that reflects reward and punishment. When working with multi-agents, it becomes harder to clearly distinguish how each agent's actions contribute to a collective reward function. This problem becomes even harder as we increase the number of agents, which can make the joint action space unrealizable, thus leading to scalability problems.

If the global reward function can be decomposed, where the environment can be partially or fully observed by agents, then each agent can locally optimize its reward function and learn its own policy using information from its own environment and information coming from neighboring agents, which are treated as part of the environment. This approach is called Independent Q-Learning (IQL) [28].

IQL is generally ideal for partially observable environments, since, through building, it learns about decentralized policies as the behavior of each agent conditions only on its own observations. IQL can cause some problems when it comes to convergence, as for each agent, the environment can be non-stationary as every other agent is also trying to learn. Research [28] shows, however, that this is not a major problem and IQL usually successfully converges to a solution.

In this work, we used IQL as a way to let agents optimize their own reward functions, but they can get information about traffic flow from adjacent intersections. To achieve this, each agent knows its adjacent intersections and it receives data from them. For example, in Figure 3 intersection B gets traffic information from intersections A, C, D and E, while intersection A gets information from intersection B. In our implementation, we used MPI (Message Passing Interface) [29] to let each agent run as its own process, and let these processes share information by using the Allgather function, which is used to gather data from all tasks and distribute the combined data to all tasks. Each agent takes only the data that corresponds to its own adjacency list.

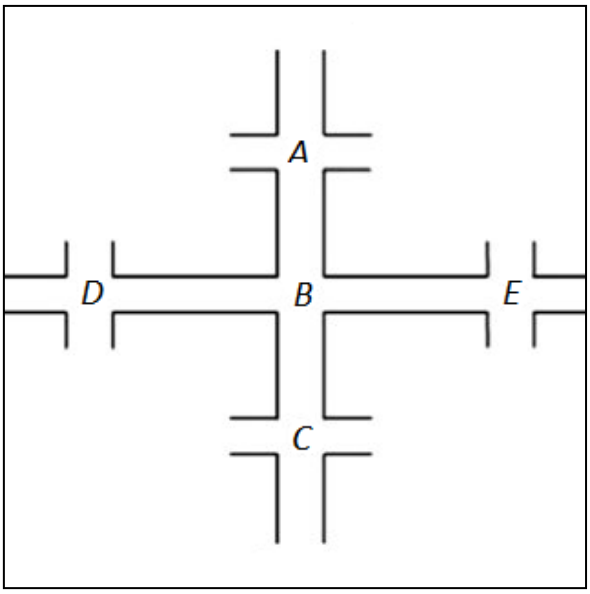

Figure 3: Multi-intersection example

3.3. States

In this work, we keep track of vehicles by using a matrix of cells that each store a vehicle ID if one is present in that cell. Each vehicle is represented by an object that keeps track of the vehicle's ID, speed (an integer value, denoting the vehicle's current speed in miles per hour) and position (cell indexes). An agent has access to the cell matrix and car information in the intersection for which it is responsible.

3.4. Action Space

In our research, we use the traffic lights at intersections to control traffic flow, so in this case, they represent the agents. The agents have the ability to show one of three different colors, green, yellow, or red at different intervals, and these are considered the actions the agent will take. These actions will be assessed by the reward function and will be scored as we will describe later.

However, agents cannot just select any color they want at any time. Within an intersection, you cannot have two traffic lights open green if they are in conflict, so these traffic lights need to be synchronized to avoid any crashes. In order to avoid this, we coordinate things in time phases, where we have the intersection represented by agents that control these phases, when and how long each phase should be opened or closed.

The representation of the environment in this research is shown in Figure 4. A 4-way intersection has multiple lanes on each road leading to the intersection. Each road is a two-way road leading in and out of the intersection. Each road has four lanes. Right turns are allowed at any time, and here we consider only car traffic, so there will be no representation of pedestrians. In this work, a traffic light is assigned to each road, where all roads at an intersection are controlled by the same agent. As mentioned before, to avoid any collisions, the agent will follow a coordinated traffic process, as shown in Figure 5.

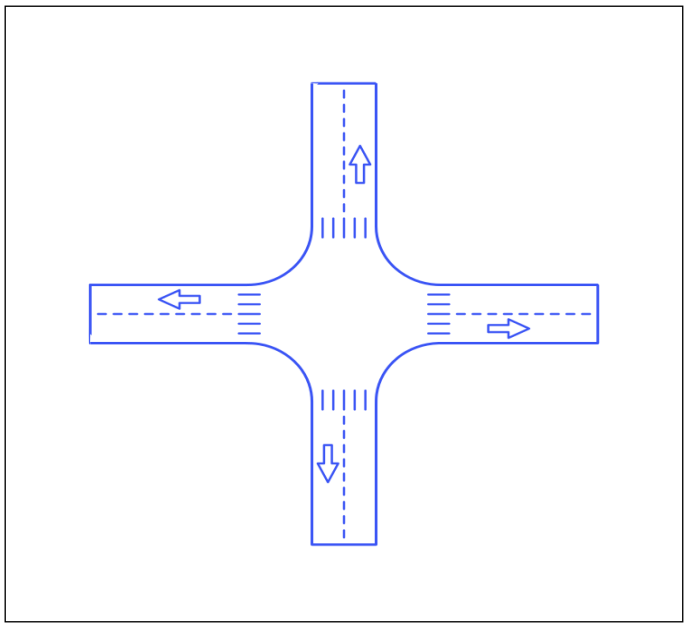

Figure 4: 4-way intersection that is represented by an agent

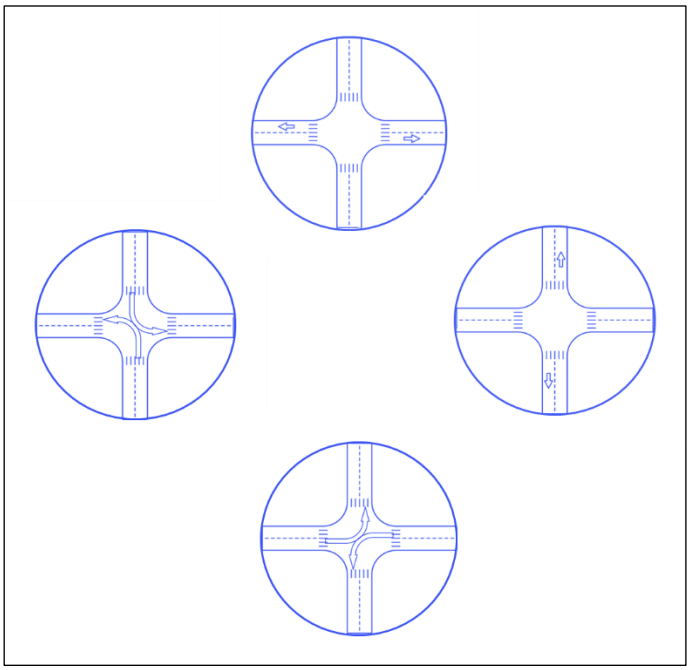

Figure 5: Actions that can be taken by agents

3.5. Reward function

As one of the main components of reinforcement learning, the reward function's main role is providing feedback about the previous actions that were taken by the agents. The reward function is a key element in the learning. In our work, the criteria that the reward function considers are:

1. Queue length: The number of vehicles standing in the queue.

2. Average waiting time in the queue.

3. Throughput: The total number of vehicles exiting the intersection every second.

4. Average speed of vehicles when exiting the intersection.

5. Average speed of vehicles when arriving at the intersection.

The last one is interesting as it can give us an indication about the flow coming from other intersections.

3.6. Reinforcement Learning Model

Q-learning is about evaluating how good an action (a) is at a certain state(s). In Q-learning, we can build a table Q[s, a] to store Q-values for all possible combinations of s and a. Then, we can sample an action from a current state and generate the reward value, which can then be used to see if we can get into a new state with the next action. In equation 1, Q-value is defined as:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma . max_A Q(s_{t+1}, a_t) - Q(s_t, a_t)) \quad (1)$$

In this equation, the current Q-value is updated using the learning rate α multiplied with the term that represents the reward value, the next Q-value and *maxA* (most valuable action among all actions in state st+1). The γ value (between 0 and 1) is used as a way to put more importance on the immediate reward versus any future reward.

One major challenge in RL is that the state space can be large enough to have all pairs of state, action evaluated. To solve this, neural networks can be used to approximate the Q-learning function. As shown in Figure 6, the NN will take the state as input and will have the possible actions the agent can take as output. When training the NN, the target will be finding the Q-values for actions and the input will be the current state of the agent.

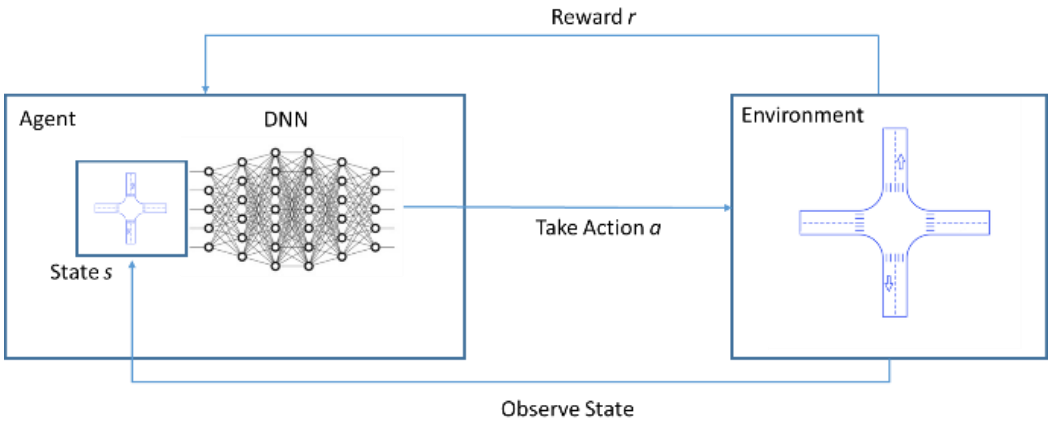

Figure 6: Deep learning RL in this work

The goal of the agent in basic Q-learning is to always choose the best action in any given state. The premise behind this choice is that the best action has the highest expected / estimated Q-value. In the beginning, however, the agent knows nothing about the environment; at first it must estimate *Q(s, a)*. The Q-values will be updated in each subsequent iteration, and will have noise, and the maximum Q-value may not be the best one, resulting in a problem called overestimation of action value (Q-value). As a result, the learning process will be very complicated and messy.

Double DQN networks have been introduced to resolve the overestimation of Q-values in DQNs. Double DQN uses two similar neural networks. One learns during the replay phase, much as DQN does, and the other is a replica of the last episode of the first model. In fact, this second model calculates the Q-value. In DQN, the Q-value is determined by applying the maximum Q-value to the next state. Obviously, if each time the Q-value calculates a high number for a given state, the value obtained from the performance of the neural network for that particular state will be higher each time. Each output neuron value will be higher and higher until the gap between each output value is high. Then what should be done to bring down the gap between the performance values (actions)? Use a secondary model that is a clone of the main model from the last episode and obviously, because the difference between the values of the second model is smaller than the main model, this second model is used to obtain the Q-value.

3.7. Traffic Generation

To test the framework against traffic, we use two car generating modes, steady vehicle flow and variable flow. The steady vehicle flow generates a steady number of cars during episodes. The variable flow tries to mimic reality by using a Rayleigh distribution with $\sigma = 2$. In the Rayleigh distribution we used, the car flow rises until a certain point, then starts to decline.

We also made runs with different traffic flows, such as high-volume traffic (5000 cars/hour) and low-volume traffic (500 cars/hour). Within each episode, we considered different ratios of how many cars go in each direction.

To simulate and generate traffic for this research, we chose the Simulation of Urban MObility (SUMO) engine. SUMO enables the simulation of intermodal transport networks, including road vehicles, public transportation, and pedestrians. SUMO provides a range of support resources that manage tasks such as route discovery, analysis, network import and emission estimation. SUMO can be improved with customized models and offers various APIs for remote simulation control.

3.8. Training

In the training part for our method, we used Keras [30] as a training framework as it is straightforward to set up easily configured to work with a GPU. The training network parameters are as shown in Table 1:

Table 1. Model Parameters.

Parameter	**Value**
No. of layers	4
Inner layers width	400
Epochs	800
Batch Size	100
Optimizer	ADAM
Loss function	Mean Squared Error
Activation function for inner layers	RELU

4. EXPERIMENTS, RESULTS AND DISCUSSIONS

4.1. Experiment specifications

In all experiments, we used a PC with the specifications shown in Table 2:

Table 2. Specifications

CPU	Intel® Xeon® 8 cores (2.6 GHz)
GPU	NVidia® Quadro® P4000 (8 GB RAM)
Memory	30 GB

In all experiments, we ran the framework for 300 episodes. 75% of simulated cars will go straight ahead. The rest of the cars will either go right or left at the intersection. We ran 5 agents, as we got 5 PCs with the same hardware specifications in Table 2. The runtime of each agent was comparable as they run on different hardware. Agents synchronize after each episode is done. Our framework is scalable, so we can run as many agents as we want as long as there is extra hardware.

4.2. Evaluation Metric

To evaluate the performance of different configurations, we used the following metrics:

1. Average delay in seconds.

2. Average queue length measured in number of vehicles in queues.

3. Average car speed measured in meters per second.

Having higher delays and longer queue lengths means less efficient signal timings. On the other hand, higher car speeds mean that vehicle flow is better. In our experiments, we generated averages for these metrics alongside the peak values.

4.3. Compared Configurations

In this work, we evaluated the following configurations based on the metrics we described:

1. DDQN + Literature Experience Replay:

This configuration uses DDQN and literature experience replay. This combination is widely used in many RL applications.

2. DQN + HER:

This configuration replaces the literature experience replay with HER. We use this to study the effect of using HER, as many publications report interesting gains from using it.

3. DDQN + HER

The final configuration in our test combines DDQN and HER to see if we can get better results compared to the other configurations.

We evaluated these configurations during different traffic situations of the day, such as peak traffic (5000 cars/hour) and off-peak traffic (500 cars/hour).

4.4. Performance evaluations

In this set of experiments, the aim is to see if the proposed reward function will lead to less cumulative waiting time for cars going through that intersection. In this set of experiments, we used HER as the experience replay engine. In Figure 7 we can see that the proposed DDQN results

in less cumulative delay for peak traffic compared to DQN. Both are using HER for replay experience.

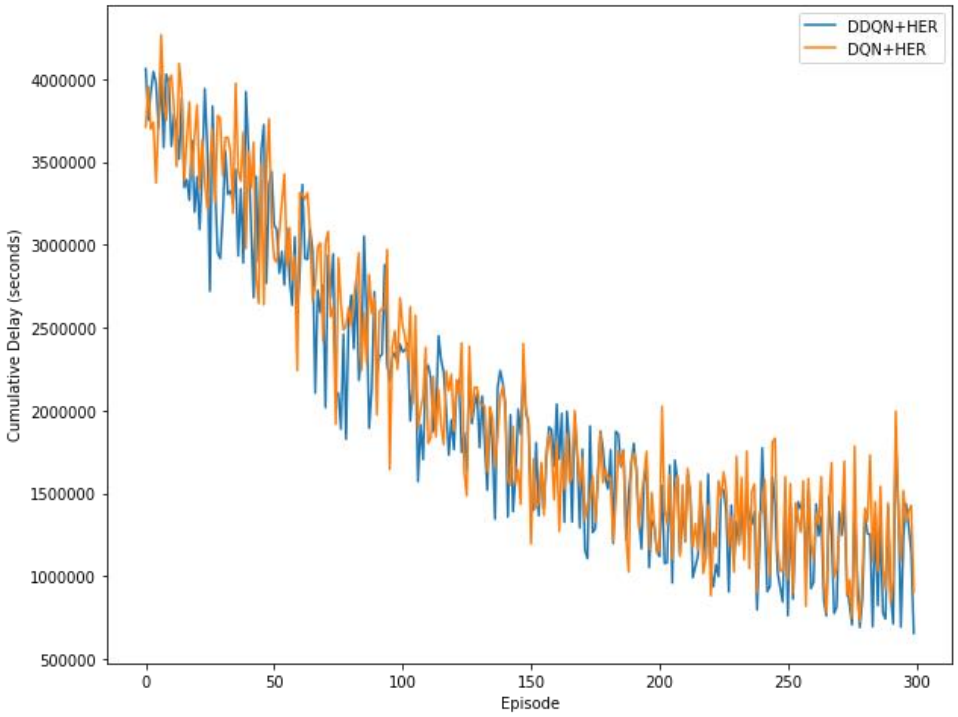

Figure 7: DDQN+HER vs. DQN+HER Cumulative Delay in seconds for 300 episodes during peak traffic.

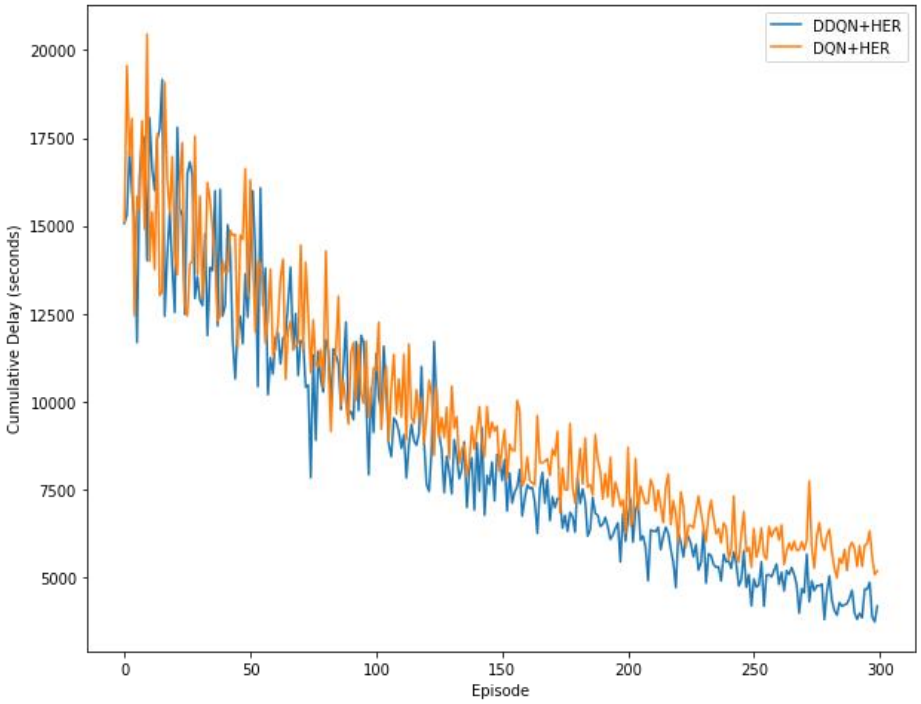

Figure 8: DDQN+HER vs. DQN+HER Cumulative Delay in seconds for 300 episodes during off-peak traffic.

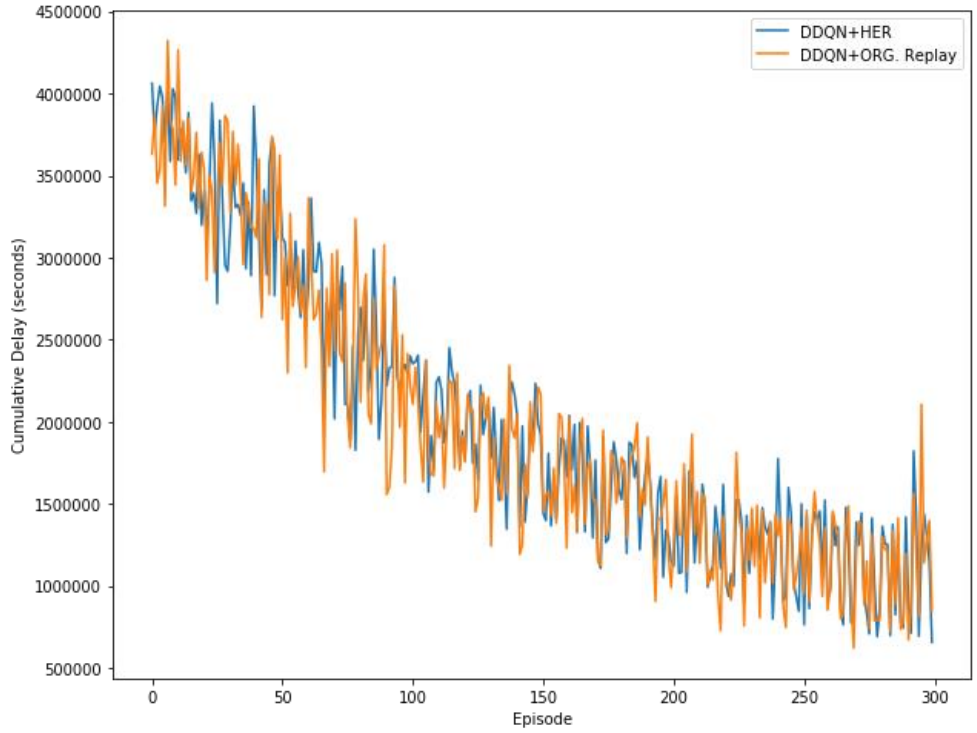

Figure 9: DDQN+HER vs. DDQN+ Original Experience replay Cumulative Delay in seconds for 300 episodes during peak traffic.

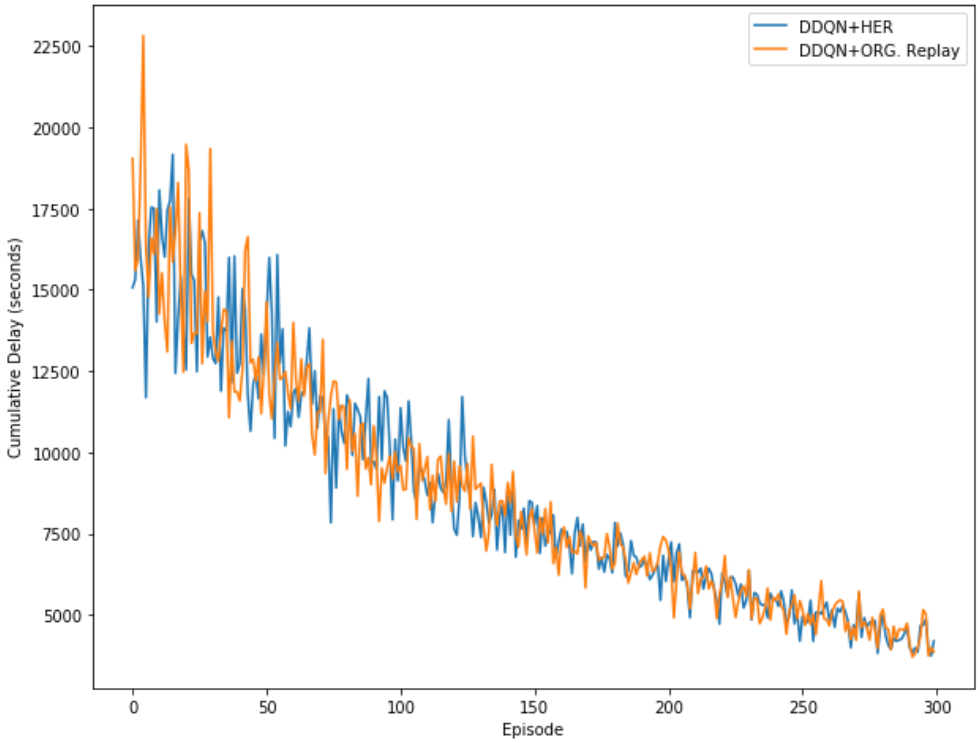

Figure 10: DDQN+HER vs. DDQN+ Original Experience replay Cumulative Delay in seconds for 300 episodes during off-peak hours.

In the next set of experiments, we tested the same setting as the previous set of experiments on off-peak traffic. In Figure 8, we can see clearly that DDQN+HER has less cumulative delay.

We also conducted a set of experiments to show how HER can improve the cumulative delay time. In Figure 9, we saw slightly better cumulative delay when running HER against literature replay experience during peak traffic. We saw comparable delay times during off-peak traffic experiment as shown in Figure 10.

Tables 3 through 6 show the different metrics we evaluated for the different configurations. We can see that the average car speed is highest for DDQN + HER in tables 3 and 4. We also observe that DDQN + HER has the least average queue length and the least average delay. When evaluating HER, we can see that DDQN gives us better results compared to using the literature replay experience. If we compare just DQN vs. DDQN, we can see that DDQN gave us better results.

We can see that the performance gains are larger when we are in peak traffic hours, but we achieved better results with DDQN + HER compared to the other configurations even in off-peak traffic.

Table 3. Models Performance Off Peak

	DQN+HER	**DDQN+RE**	**DDQN+HER**
Average Car Speed (m/s)	4.285	4.526	5.31
Average Queue Length (car)	1.763	1.686	1.604
Average Delay (s)	9523.156	91071.86	8665.65

Table 4. Models Performance Peak

	DQN+HER	**DDQN+RE**	**DDQN+HER**
Average Car Speed (m/s)	2.47	3.69	4.1
Average Queue Length (car)	384.59	375.55	369.95
Average Delay (s)	2076829.38	1973975.79	1997730.9

Table 5. Models Performance Off Peak

	DQN+HER	**DDQN+RE**	**DDQN+HER**
Max Car Speed (m/s)	5.71	5.6398	7.3
Max Queue Length (car)	6.3	4.1	3.5485
Max Delay (s)	20449	460613	19162

Table 6. Models Performance Peak

	DQN+HER	**DDQN+RE**	**DDQN+HER**
Max Car Speed (m/s)	3.521	4.1175	4.434
Max Queue Length (car)	815.3	800.41	752.23
Max Delay (s)	4265812	4322216	4062073

5. CONCLUSIONS

In this paper, we introduced a framework that will optimize traffic signal times using multi-agent Deep MARL and Hindsight Experience Replay (HER). The framework enables multiple agents to run simultaneously, each optimizing a certain traffic signal. Each signal shares data with adjacent agents to try to achieve better overall reduction of wait time.

We ran the multiple agents using MPI, and we used SUMO as the simulation engine for cars and traffic lights. We also designed a reward function that depends on data from the local environment and data coming from adjacent agents. Results show reduced cumulative delay time when using DDQN and HER in different settings as detailed in section 4.

In future work, we plan to expand the network to test with other deep learning techniques, such as upside-down RL, and try to use data for different times of the day to better optimize traffic signal timing. In addition, we will test the framework on intersection plans from real roads. As for the Multi agent scalability, we are planning to expand the MPI code to accommodate different hardware scenarios, such as having one PC running multiple agents at the same time.

REFERENCES

[1] J. C. Medina and R. F. Benekohal, “Traffic signal control using reinforcement learning and the max-plus algorithm as a coordinating strategy,” in Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on. IEEE, 2012, pp. 596–601.

[2] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, “Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto,” IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 3, pp. 1140–1150, 2013.

[3] M. A. Khamis and W. Gomaa, “Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework,” Engineering Applications of Artificial Intelligence, vol. 29, pp. 134–151, 2014.

[4] M. Wiering, J. Van Veenen, J. Vreeken, and A. Koopman, “Intelligent traffic light control,” Institute of Information and Computing Sciences. Utrecht University, 2004.

[5] B. Abdulhai, R. Pringle, and G. J Karakoulas. 2003. Reinforcement learning for true adaptive traffic signal control. Journal of Transportation Engineering 129, 3 (2003), 278–285.

[6] J.J. Henry, J.L. Farges and J. Tuffal. “The Prodyn Real Time Traffic Algorithm,” IFAC Proceedings Volumes, vol. 16, no. 4 pp. 305-310, 1983.

[7] Yi Hu, P. Thomas and R. J. Stonier, "Traffic signal control using fuzzy logic and evolutionary algorithms," 2007 IEEE Congress on Evolutionary Computation, Singapore, 2007, pp. 1785-1792.

[8] M. K. Tan, H. S. E. Chuo, R. K. Y. Chin, K. B. Yeo and K. T. K. Teo, "Optimization of urban traffic network signalization using genetic algorithm," 2016 IEEE Conference on Open Systems (ICOS), Langkawi, 2016, pp. 87-92.

[9] B. Yin, M. Dridi and A. El Moudni, "Comparing dynamic programming based algorithms in traffic signal control system," 2016 4th IEEE International Colloquium on Information Science and Technology (CiSt), Tangier, 2016, pp. 604-609.

[10] J. D. Kelleher, Deep Learning. MIT Press, 2019.

[11] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[12] M. Abdoos, N. Mozayani, and A. L. Bazzan, “Traffic light control in non-stationary environments based on multi agent q-learning,” in Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on. IEEE, 2011, pp. 1580–1585.

[13] H. Nguyen, H. M. La and M. Deans, "Hindsight Experience Replay With Experience Ranking," 2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob), Oslo, Norway, 2019, pp. 1-6

[14] M. Abdoos, "A Cooperative Multi-Agent System for Traffic Signal Control Using Game Theory and Reinforcement Learning," in IEEE Intelligent Transportation Systems Magazine, 2020.

[15] I. Arel, C. Liu, T. Urbanik and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," in IET Intelligent Transport Systems, vol. 4, no. 2, pp. 128-135, June 2010.

[16] J. V. S. Busch, V. Latzko, M. Reisslein and F. H. P. Fitzek, "Optimised Traffic Light Management Through Reinforcement Learning: Traffic State Agnostic Agent vs. Holistic Agent With Current V2I Traffic State Knowledge," in IEEE Open Journal of Intelligent Transportation Systems, vol. 1, pp. 201-216, 2020.

[17] T. Chu, J. Wang, L. Codecà and Z. Li, "Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control," in IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 3, pp. 1086-1095, March 2020.

[18] W. Genders, S. Razavi, "Evaluating Reinforcement Learning State Representations for Adaptive Traffic Signal Control," in International Journal of Traffic and Transportation Management, vol. 130, pp. 26-33, April 2018.

[19] M. Khamis, & W. Gomaa, "Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework," in Engineering Applications of Artificial Intelligence, vol. 29, pp. 134-151, March 2014.

[20] R. Zhang, A. Ishikawa, W. Wang, B. Striner and O. K. Tonguz, "Using Reinforcement Learning With Partial Vehicle Detection for Intelligent Traffic Signal Control," in IEEE Transactions on Intelligent Transportation Systems, pp. 1-12, March 2020.

[21] "Simulation of Urban Mobility SUMO" [Online]. Available: https://www.eclipse.org/sumo/. [Accessed Jan 2020].

[22] X. Liang, X. Du, G. Wang and Z. Han, "A Deep Reinforcement Learning Network for Traffic Light Cycle Control," in IEEE Transactions on Vehicular Technology, vol. 68, no. 2, pp. 1243-1253, Feb. 2019.

[23] L. Li, Y. Lv and F. Wang, "Traffic signal timing via deep reinforcement learning," in IEEE/CAA Journal of Automatica Sinica, vol. 3, no. 3, pp. 247-254, 10 July 2016.

[24] W. Genders & S. Razavi, “Using a deep reinforcement learning agent for traffic signal control,” arXiv preprint arXiv:1611.01142, November 2016.

[25] S. S. Mousavi, M. Schukat and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," in IET Intelligent Transport Systems, vol. 11, no. 7, pp. 417-423, September 2017.

[26] M. Wiering, “Multi-agent reinforcement learning for traffic light control,” in ICML, pp. 1151–1158, 2000.

[27] M. Abdoos, N. Mozayani, & A.L.C Bazzan, "Holonic multi-agent system for traffic signals control. Eng," in Engineering Applications of Artificial Intelligence, vol. 26, no. 6, pp 1575-1587, May–June 2017.

[28] X. Gong, B. Ding, J. Xu, H. Wang, X. Zhou and H. Jia, "Synchronous n-Step Method for Independent Q-Learning in Multi-Agent Deep Reinforcement Learning," 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Leicester, United Kingdom, 2019, pp. 460-46.

[29] "Message Passing Interface Tutorial" [Online]. Available: https://mpitutorial.com/. [Accessed Jan 2020].

[30] "Keras" [Online]. Available: https://keras.io/. [Accessed Jan 2019].

Authors

Mrs. Areej Salaymeh is a Ph.D. candidate at the department of computer science in Wayne State University. she worked in the field of High-Performance Computing, Artificial Intelligence, Machine Learning, and Intelligent Transportation Systems. Mrs. Areej worked as a full time Lecturer at the Department of Computer Science in Wayne State University between 2017 and 2020. Now Mrs. Areej works as a Assistant Professor in the college of Business and Information Technology at Lawrence Technological University.

Dr. Loren Schwiebert is an Associate Professor in the Department of Computer Science at Wayne State University Detroit, MI. He received the B.S. degree in computer science (with a dual major in mathematics) from Heidelberg College, Tiffin, OH, and the M.S. and Ph.D. degrees in computer and information science from the Ohio State University, Columbus. His research interest in machine learning, General-Purpose Programming on Graphics Processors, Scientific Computing on many-core and multi-core systems.

Dr. Stephen Remias is an Assistant professor in the Civil and Environmental Engineering department at Wayne State University Detroit, MI. He received his B.S. degree in Civil Engineering from Michigan State University in 2009. He received his M.S. and Ph.D. degrees in Civil Engineering from Purdue University in 2014. His research interests in Intelligent Transportation Systems, Traffic Operations, Connected Infrastructure, and Safety and Mobility Performance Measures