Comparison of Q-Learning based Traffic Light Control Methods and Objective Functions

Péter Pálos
Dept. of Networked Systems and Services
Budapest University of Technology and Economics
Budapest, Hungary
peterpalos@edu.bme.hu

Árpád Huszák
Dept. of Networked Systems and Services
Budapest University of Technology and Economics
Budapest, Hungary
huszak@hit.bme.hu

***Abstract*—Traffic control is a cardinal issue in the life of urban areas. The traditional fixed duration traffic signal methods do not provide the most optimal solution while the volume of vehicles is constantly growing. Reinforcement learning is a promising approach for adaptive signal control that observe, learn, and select the optimal traffic light control action. In this paper, we present a comparison of Deep Reinforcement Learning based traffic optimization methods. This alternative way let the controllers to learn by the dynamics of the traffic and adapt to it. Our aim was to investigate the performance of advanced DQN variants in a single intersection environment. We examined four Q-Learning approaches: DQN, Double DQN, Dueling DQN and Double Dueling DQN with six different objective functions in a freely adaptable signal cycle environment. Our results show that the Double Dueling DQN based agent outperform the other models on the basis of different aspects and we can conclude about the possibilities of integration for real-life traffic management.**

Keywords—Traffic Light Control, Machine Learning, Deep Reinforcement Learning

I. INTRODUCTION

The increasing population of metropolises also results an increase in traffic density leading to traffic jams and accidents which is a serious urban management problem. Traffic optimization through the coordination of traffic lights is necessary for both economic and ecological reasons to ensure a proper urban life. Although, control systems have undergone continuous development, if the system is unable to adapt to the dynamism of the traffic demand, one of its most valuable capability is lost.

Traffic management systems were intensively studied in the last few decades. The two type of traditional traffic signal controllers are the pre-timed and actuated control systems [1], which are still widely used in urban areas. Pre-timed control systems are using predefined signal phase lengths based on historical traffic data collected from the traffic network in different times in a day with. The actuated control system dynamicity controls the traffic taking only the current and local traffic conditions into account.

The advances in information technology in the field of data storage and computing capacity allow us to take advantage of new technological solutions such as machine learning for adaptive traffic light control. Any process that generates a large amount of data is suitable for this purpose and with the growth in the number of traffic cameras and detectors a huge raw traffic data is produced day by day without making full use of it. The successfulness of machine learning in handling raw data and solving complex problems has been widely investigated in recent years.

Reinforcement learning is a promising approach to be used for traffic light control purposes. The main idea behind it is to interact with the environment, observe the changes and estimate how good the taken action was. Number of reinforcement algorithms were developed, but the most popular methods are Q-learning and Deep Q-Network (DQN) used in case of large observation and action spaces. In most of the cases, in the research papers only one type of algorithm is analyzed with a single reward function.

In this work, our aim is to provide a novel comparison of the performance of reinforcement learning models and different reward functions in the area of traffic management problems where the agent is free to design the signal cycles and the self-generated traffic demand prepares them for any eventuality. We analyzed the performance of DQN and its variants, Double DQN, Dueling DQN and Double Dueling DQN for traffic light control in a single intersection environment. We selected six objective calculation methods shown to be successful in adaptive adjustment of signal phases, based on previous studies. In our experiments we analyzed the performance of the four DQN approaches from waiting time, number of halting vehicles and CO_2 exhaustion point of view.

The structure of the paper is as follows. In Section II. we provide a brief overview of previous reinforcement learning based methods used in traffic management. The used simulation environment and the reward functions, as well as detailed theoretical backgrounds of the selected methods are introduced in Section III. In Section IV. the training parameters of the DQN algorithms and the comparison metrics are described. The obtained results are also presented and evaluated in the same section. In the final section we summarize and conclude the paper.

II. RELATED WORKS

According to Gartner, Stamatiadis and Tarnoff [2] there are five levels of intelligent decision-making in traffic control.

- The First Level Control contains previously calculated timing plans from historical data. The operator determines the suitable plan mainly on the basis of time of the day but also can select directly.
- The Second Level Control computes real-time and defines the suitable timing plan based on traffic demand.
- Third Level Control is similar to the second one with the difference that can change plans at a shorter time interval.
- Fourth Level Control allows the integration of several subsystems

- The Fifth Level Control operates on the basis of a self-learning algorithm, learns and adapts to traffic dynamism of real-time data.

In classical signal control schemes the Webster method by Koonce et al. [3] is generally used to minimize the travel time of all vehicles in a single intersection by defining the cycle length and phase split based on Webster's Equation. It is a widely-used method which takes into account several attributions like number of phases, acceleration, deceleration, demand fluctuation. GreenWave developed by Roess, Prassas and McShane [4] reduces the number of stops for traveling along one certain direction by calculating the offsets between intersections. Self-Organizing Traffic Light Control (SOTL) by Cools, Gershenson, and D'Hooghe [5] is based on requests on the current phase. The request to extend the green signal time is generated when the duration does not reach the specified minimum value and the request to change the phase when the number of waiting vehicles is larger than a threshold. Max-pressure by Varaiya [6] aims to lower the „pressure" of the phases, which is defined as the number of vehicles on incoming lanes minus the number of vehicles on the outgoing lanes.

Machine learning based traffic signal control schemes are intensively investigated solutions with a focus on reinforcement learning algorithms in most of the cases. Reinforcement learning is a goal-oriented machine learning technique, which learn how to attain a complex objective or how to maximize along a particular dimension over many steps. The approach is concerned with how the agents should take actions in an environment and maximize the overall reward gained as Fig. 1. illustrates.

Fig. 1. Reinforcement Learning cyclicality

In case of a traffic control scheme the environment is monitored by detectors (e.g., induction loop, magnetic sensor, traffic camera image processing), determining the state of the environment. The action in this case is the traffic light control decision, such as prolong or shorten the green phase, switch to another control program, etc.

A. Q-learning

Q-learning [7] is one of the most popular model-free reinforcement learning algorithm investigated for traffic light control due to its simplicity and performance. The agent calculates the effectiveness of each action in simulation states with the Q-function:

$$Q^{new}(s_t, a_t) = Q(s_t, a_t) + \\ + \alpha \cdot (R_t + \gamma \cdot maxQ(s_{t+1}, a) - Q(s_t, a_t)) \quad (1)$$

where $Q(s_t, a_t)$ is a given Q-value for t time step, α is the learning rate, R_t the earned reward and γ is the discount factor. Each state action pair with the corresponding Q-value are stored in the Q-table. The method has been widely investigated in field of traffic signal control [8][9][10][11]. Due to technical limitation, the state space should be of low complexity. With combination of Q-learning and Deep Learning to estimate the Q-values, the system is much more flexible in terms of defining the state space of traffic [12]. Dueling DQN [13] was used for signal control by Liang and Du [14]. Van der Pol [15] uses Double DQN developed by Van Hasselt, Guez and Silver [16]. Wang, Xie, Huang, Zeng and Cai present the usability of the mixture of Double DQN and Dueling DQN (3DQN) in signal management [17].

B. Deep Q-Network

As the included simulation parameters become more complex, the state space increases and the memory and computational requirements for calculating with them grow excessively high. To avoid the problem a combined version of Q-learning and Machine Learning is developed, what is called Deep Q-Network (DQN).

In the case of DQN the cyclicality of the agent and the environment is the same as in regular Q-learning (Fig. 1.), however instead of storing Q-values in a table we estimate them using a deep neural network. Q-values are updated with a similar Q-function shown in equation (1)

$$Q(s_t, a_t) = R_t + \gamma \cdot maxQ(s_{t+1}, a'), \quad (2)$$

where $Q(s_t, a_t)$ is the prediction of the neural network based on the state space, $Q(s_{t+1}, a')$ is the prediction of the same neural network for the following state space, and gamma is the associated discount factor. The updated Q-values will be the target values in training of the neural network, and their inputs will be the state spaces. To stabilize the learning a so-called Replay Buffer is created where the current state space, action, reward and the next state space is stored continuously. The size of the Replay Buffer can be freely determined and during training mini-batches are created from it by random sampling.

C. Double DQN

The target value is obtained as the sum of the reward and the highest Q-value of the next state, which is estimated by the same neural network. If the network regularly assigns a higher Q-value to non-optimal actions, the algorithm get stuck in a local optima without finding the optimal action during the training. To avoid this problem Haaselt developed Double DQN [16], which is based on periodically constant target values. A second neural network (target model, Q^c) with periodically constant weights is made with the same structure as the base neural network. The learning is frozen constantly and the weights from the original model are copied after certain number of steps. The formula for calculating the target value is as follows:

$$Q(s_t, a_t) = R_t + \lambda \cdot Q^c(s_{t+1}, \arg\max Q(s_{t+1}, a')) \quad (3)$$

D. Dueling DQN

When in a given state all actions give the same Q-value, it assumes that none of the actions gives a more advantageous position than the others. Dueling DQN developed by Wang et al. [13] is able to filter out states in which it is not necessary to learn the value of the actions belonging to them. It is especially relevant in environments where the action does not

always affect it substantially. In traffic control it is a cardinal issue, the model must learn that it is not necessary to change the signal cycle during periods without traffic demand. The structure of the neural network is different from the previous ones because we create two separated threads at the end of the network. One estimates the Q-values in the same way as before and the other thread is the so-called advantage function, which has one output neuron and determines the advantage of choosing an action in a given state. The threads are then aggregated according to

$$Q(s,a) = V(s) + A(s,a) - \frac{1}{|A|}\sum_{a'} A(s,a') \qquad (4)$$

where $V(s)$ is the state value and $A(s, a)$ is the advantage for each action. The mechanism of Dueling DQN and Double DQN can be mixed without any problem what is called Double Dueling DQN (D3QN).

Our aim in this paper was to compare the performance of DQN variants in a single intersection environment deployed in a microscopic traffic simulator and evaluate different reward functions.

III. Simulation Environment and Objective Function Setup

In order to evaluate the performance of the variants of the DQN reinforcement learning algorithm, we deployed a single, traffic light controlled intersection in SUMO [18]. SUMO is an open source traffic simulation software with detailed configuration options and utilities. The advantage of the SUMO microscopic simulator tool is that it can be controlled in run-time through the Traffic Control Interface (TraCI). Hence, our Python-based tool handled as the traffic light control processes and the traffic state measurements as the traffic flow settings. The structure of the examined intersection with a traffic light in the center is shown in Fig. 2.

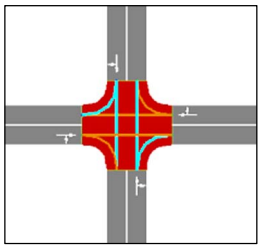

Fig. 2. Controlled intersection in SUMO

In the simulated environment, the vehicles can continue their routes straight or with a right turn. To detect traffic, detectors were deployed in distance of 50 meters from the signal head on each of the four inbound traffic-generating lanes. The data from the detectors are designed to simulate information that can be extracted from a loop detector or real camera image. We have currently developed three metrics, each ranging from 0 to 10:

- number of vehicles in the detector area,
- percentage of area occupied by vehicles,
- average speed of the vehicles in the area.

The metrics are generated from all four directions, therefore the state space forms a 4 × 3 size vector that is expanded with the current lamp cycle.

To ensure convertibility, the environment was created according to OpenAI Gym [19] standards. OpenAI is a non-profit research company and their toolkit allows developing and comparing reinforcement learning algorithms. To allow dynamic intervention the step of the custom environment is the same as the step defined by TraCI, which is one second. The agent can make two choices, extend the current signal phase or interrupt it and move on to the next one. In case of a signal phase change, the amber (yellow sign) phase automatically starts, which cannot be intervened for a period of three seconds. In case of penalty-based reward functions, the penalty is added up during the amber phase, penalizing frequent cycle changes. The reason is that the traffic can be congested during this period, because all directions are blocked.

To ensure that the agent can experience and learn from all possible traffic situations, we have created a custom traffic demand based on sine functions. The traffic demand for each route (D_i) is determined by the probability of vehicle insertion according to the following equation

$$D_i = P_{\max} \cdot \frac{\sin\left(s + i \cdot \frac{s}{10}\right) + 1}{2} \qquad (5)$$

where $P_{\max}$ determines the maximum value of the range, preventing overly dense vehicle generation, s is the current number of steps and i is the index of the current route, making the period of sine function a unique length. The vehicle placement probabilities for the four directions are as shown in Fig. 3.

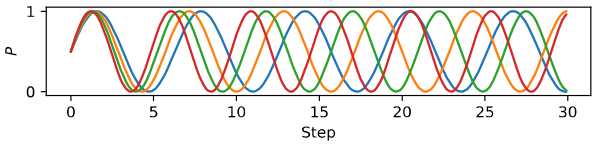

Fig. 3. Vehicle placement probabilities

The training process is iterated through hundreds of thousands of steps, the above sine function combinations will also be repeated giving the agent an opportunity to explore the demand for all possible traffic situations. To avoid overfitting problem for each step the probability of locating the vehicle is randomly selected from a value determined by the sine function and a random probability.

An important pillar of training an agent is what kind of reward it will receive for the actions taken. In our paper, we created six different reward functions to compare their effectiveness:

- WTMin: Waiting Time Minimization (a vehicle is halted when its travel speed than 0.1 m/s)
- ASMax: Average Speed Maximization [20]
- TTMin: Travel Time Minimization [21]
- CTMax: Completed Trip Maximization: the number of vehicles leaving the simulation at the end of their route [21]
- WCTMax: Weighted Completed Trip Maximization: the number of vehicles leaving the simulation at the

end of their route weighted by the length of the shortest route [21]

- TMQMax: Throughput Minus Queue Maximization: the difference between the number of vehicles starting their trip (entering the simulation) and the number of halting vehicles (slower than 0.1 m/s) [21]

To implement the Neural Network based agent in DQN models we used the Keras library [22]. The architecture of the network is as follows: the input shape dynamically adjusts to the environment's observation space, which means 13 neurons in the current environment. The input layers connects with three hidden Dense layers containing 512, 256 and 64 neurons in order. All hidden layers come with Rectified Linear Unit (ReLU) activation function. In the case of Dueling DQN, after the last hidden layer the forward propagation splits into two branches, one with a single neuron and one with the number of possible actions in the environment, which means two neurons in this case. The output of layers do not go through an activation function. The two branches are then aggregated according to equation (4). In the simple DQN architecture instead of separation, there is an output layer containing two neurons with linear activation function. The weights are initialized with He Normal Initialization [23] in all of the layers.

IV. Simulation Results

To test the trained models, we created a 1000 step long process with a traffic demand completely different from the sinusoidal function used during training. The traffic demand for testing was generated by the SUMO's randomTrips.py script to protect against the distortion of overfitting and to ensure comparability.

Performance evaluation of DQN based models and objective functions was performed based on four metrics, which are the reward earning dynamics during training, overall waiting time in the network, overall number of halting vehicles in the network and the overall CO^2 emission produced by vehicles in the network during testing. Below we highlight the most important results with TensorBoard based visualization and data smoothing for easier interpretation. The graphs do not show the reward values up to 100,000 steps which is not an error. According to the environment description, the steps during yellow cycles do not appear in the environment and causes non-uniform line lengths, thus also providing extra information about the number of yellow cycles.

TABLE I. PARAMETER SETTINGS USED IN THE COMPARISONS

Parameter settings	
Parameter	***Value***
Starting value of epsilon greedy policy	1.0
Discount factor of epsilon greedy policy	0.99
Minimum epsilon value	0.02
Discount factor of Q-value update	0.9
Learning rate of the Neural Network	0.0005
Batch size during training	32
Target model weight updating frequency in Double DQN	500
Frequency of training	1
Number of steps before starting learning	1000
Size of the Replay Buffer	50,000
Maximum probability of vehicle generating	0.1
Number of steps during training	100,000

The operation of the reinforcement learning algorithms is influenced by a number of parameters. The parameter settings used in the comparison are shown in Table I.

In the first part of the comparison, we examined the reward dynamics of different DQN variants. Fig. 4. shows the progress of reward earning of different DQN models trained on average speed maximization (ASMax). Based on the observation of the trajectory, it can be stated that the different models learned to almost the same level of maximal reward during the training period, however there are differences in the dynamics of learning. According to the figure the simple DQN needed the most steps to reach a stable reward level and at the fastest pace the D3QN was able to achieve it.

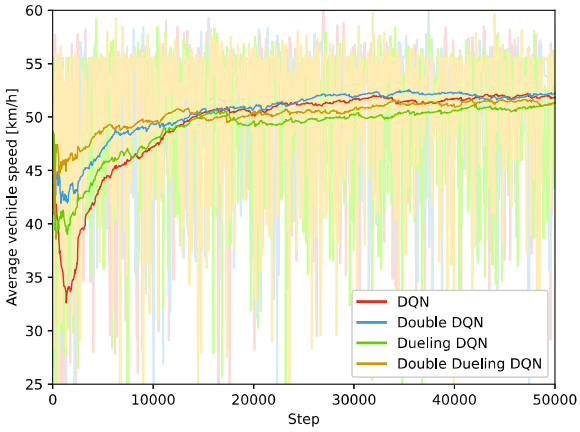

Fig. 4. Reward values (average speed maximization) during training

In the case of the models trained on the basis of maximizing the weighted completed trips (WCTMax) what is shown in Fig. 5, there are less clear differences and the dynamics are noisier, which is presumably due to the greater vulnerability to traffic demand. Nevertheless, it can be observed that the simple DQN shows a slower pace of learning in this case as well, and the stable level remains below the other variants. Dueling DQN, Double DQN and D3QN provided nearly similar performance except for periodic oscillations.

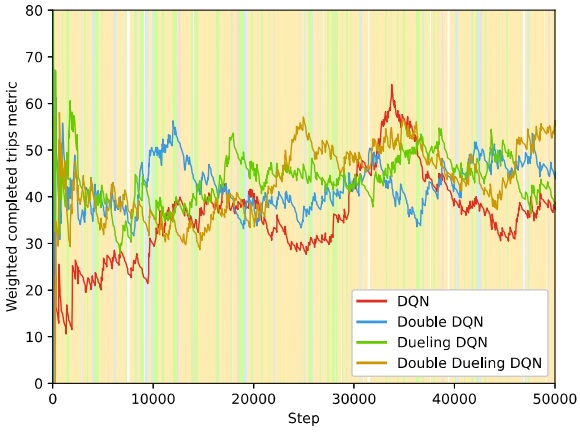

Fig. 5. Reward values (weighted completed trips) during training

While the benefits of different DQN models are most noticeable in the learning dynamics of maximizing rewards, they can also improve agent performance in other ways, which we can analyze based on testing measurements.

In the detailed tests, DQN and Dueling DQN provided the least stable results, with multiple and large oscillations.

Double DQN and D3QN produced stable and low values both for overall waiting time (Fig. 6.), number of halting vehicles (Fig. 8.), and CO^2 emission (Fig. 9.), therefore the comparison of reward functions will be presented using D3QN models for clarity.

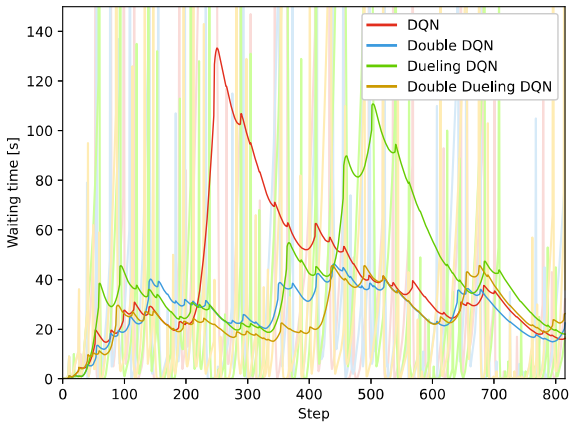

Fig. 6. Overall waiting times during testing the trained model

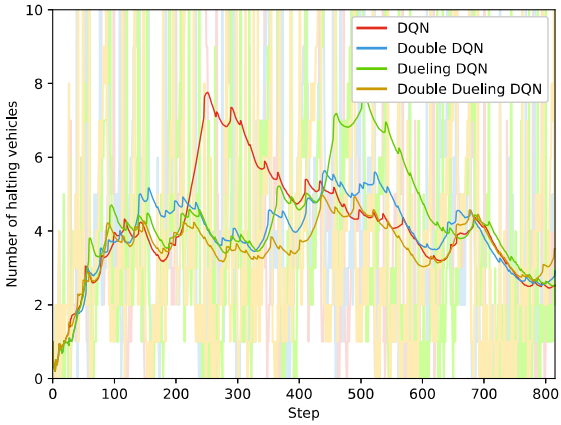

Fig. 7. Number of halting vehicles during testing the trained model

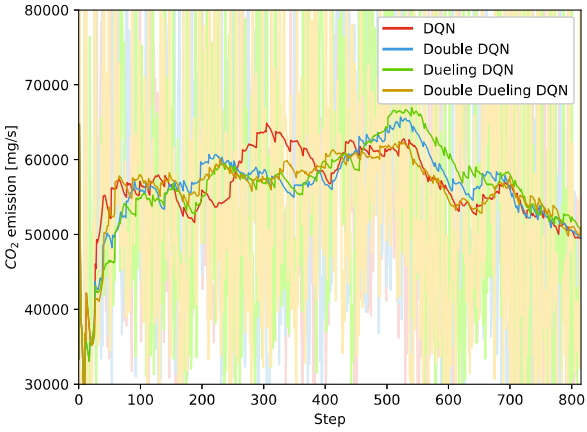

Fig. 8. CO_2 emission during testing the trained model

The second half of the comparison shows the visual pattern of the measured variables for the different reward functions. Although the average speed based objective function generates a low latency during the test as we can see on Fig. 9., the number of steps shows that it operates with many amber (yellow sign) phases.

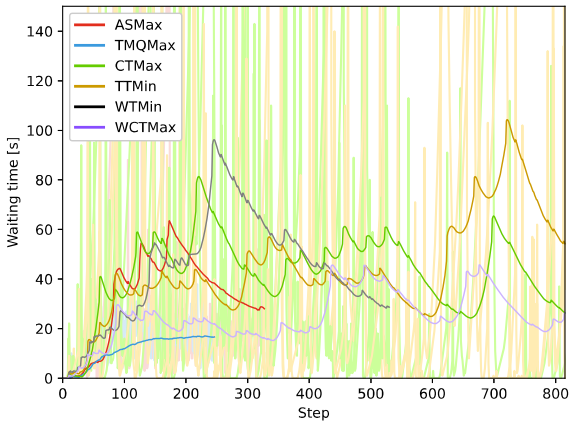

Fig. 9. Comparison of reward functions from the waiting time point of view during testing the trained model

Testing on the visual GUI interface also supports this, the model decides to change the lamp phase in almost every step. The same is true for the TMQ base reward function and with a less extent for the waiting time base reward function. Weighted trip number maximization appears to be more effective than its unweighted counterpart and produced the most stable and lowest values over the entire test. When minimizing travel time, the values are low, but at the same time contain large fluctuations.

On Fig. 10. the problem of TMQMax and average speed based (ASMax) reward functions is even more pronounced in the case of CO_2 emission. The lowest CO_2 emission was achieved by weighted trip maximization and its unweighted counterpart resulted in nearly similar values. Travel time minimization (TTMin) as in the previous graph sometimes shows lower values, but also contains strong fluctuations.

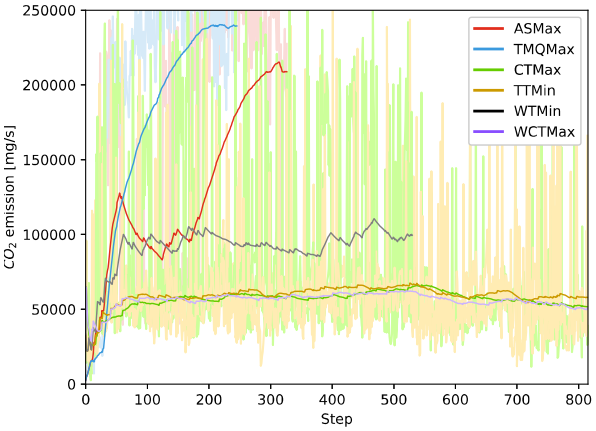

Fig. 10. Comparison of reward functions from the CO_2 emission point of view during testing the trained model

The number of waiting vehicles is the third and the final part of the comparison, the results can be seen on Fig. 11. The trajectory supports the conclusions so far, weighted trip maximization seems to be the most appropriate reward function in terms of number of waiting vehicles. Its unweighted counterpart involves a slightly higher vehicle number while it is even more characteristic in case of travel time (TTMin) and waiting time (WTMin) base minimization functions.

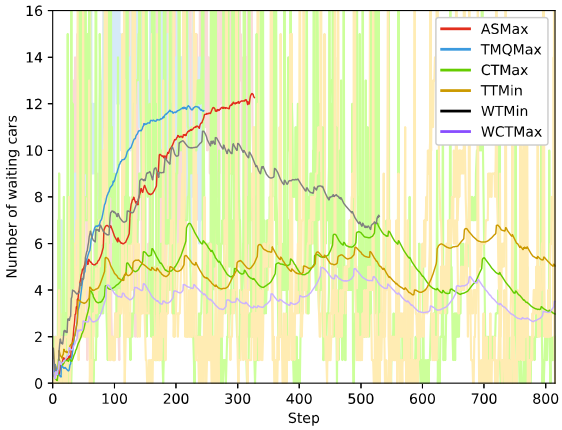

Fig. 11. Comparison of reward functions from the number of waiting vehicles point of view during testing the trained model

The performance of several DQN algorithms and objective functions were analyzed and compared in this work. The summarized results are introduced in Table II.

TABLE II. SUMMARY OF DQN MODELS AND REWARD FUNCTIONS

Numerical evaluation (averages)			
	CO_2 emission (mg/s)	***Halting vehicle number***	***Waiting time (s)***
DQN	56252	4.126	39.97
Dueling DQN	56846	4.222	40.31
Double DQN	56376	3.926	27.75
D3QN	**56181**	**3.645**	**26.33**
ASMax	162900	10.23	34.52
TMQMax	213633	10.51	**14.81**
CTMax	56449	4.548	41.90
WCTMax	**56181**	**3.645**	26.33
TTMin	60229	5.127	47.77
WTMin	95474	8.264	44.49

CONCLUSIONS

In this work four DQN variant reinforcement learning methods (Deep Q-Network, Double DQN, Dueling DQN and Double Dueling DQN) and the six objective functions (waiting time minimization, average speed maximization, travel time minimization, completed trip maximization, weighted completed trip maximization and throughput minus queue maximization) were investigated and evaluated. We concluded that the learning dynamics of the D3QN model represent a major advance over the simpler DQN variants as shown in Table II., so it can be used well to train additional routing agents. The benefits of the Dueling DQN architecture is crucial to avoid signal cycle changes during traffic-free periods. Based on the three testing measurements which are presented on graphs and in Table II. weighted completed trip number maximization appears to be the most effective reward function. It also allocates traffic properly according to the visual GUI interface, in the case of high traffic demand act in a similar way to the currently used fix period length cycle settings, but in the case of low traffic, it makes faster phase changes by taking into account the arriving vehicles. Based on the results we concluded that a fully machine learning based traffic management can be unpredictable in some cases, which can result in serious traffic problems that are not acceptable from a security perspective. In spite of that, it can be effective and justified to complement traditional methods for understanding traffic patterns and for dynamic intervention.

REFERENCES

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

[2] N. H. Gartner, C. Stamatiadis and P. J. Tarnoff, "Development of Advanced Traffic Signal Control Strategies for Intelligent Transportation Systems: Multilevel Design," TRANSPORTATION RESEARCH RECORD, pp. 98-105, 1494.

[3] P. Koonce, L. Rodegerdts, K. Lee, S. Quayle, S. Beaird, C. Braud, J. Bonneson, P. Tarnoff and T. Urbanik, "Traffic Signal Timing Manual," United States. Federal Highway Administration, 2008.

[4] R. P. Roess, E. S. Prassas and W. R. McShane, Traffic engineering, Prentice Hall, 2004.

[5] S.-B. Cools,, C. Gershenson, and B. D’Hooghe., "Self-organizing traffic lights: A realistic simulation," Advances in applied self-organizing, pp. 45-55, 2013.

[6] P. Varaiya, "The max-pressure controller for arbitrary networks of signalized intersections," in Advances in Dynamic Network Modeling in Complex Transportation Systems, Springer, 2013, pp. 27-66.

[7] C. Watkins, "Learning from Delayed Rewards," Cambridge University, 1989.

[8] M. Abdoos, N. Mozayani and A. L. C. Bazzan, "Traffic light control in non-stationary environments based on multi agent q-learning," in 14th International IEEE Conference on Intelligent Transportation Systems, 2011.

[9] S. Araghi, A. Khosravi, M. Johnstone and D. Creighton, "Q-learning method for controlling traffic signal phase time in a single intersection," in 16th International Conference on Intelligent Transportation Systems, 2013.

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

[11] D. Zhao, Y. Dai and Z. Zhang, "Computational intelligence in urban traffic signal control: A survey," Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, pp. 485-494, 2012.

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

[13] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot and N. de Freitas, "Dueling network architectures for deep reinforcement learning," arXiv, 2015.

[14] X. Liang and X. Du, "Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks," arXiv, 2018.

[15] E. van der Pol, "Deep reinforcement learning for coordination in traffic light control," University of Amsterdam, 2016.

[16] H. van Hasselt, A. Guez and D. Silver, "Deep reinforcement learning with double q-learning," in Conference on Artificial Intelligence, 2016.

[17] S. Wang, X. Xie, K. Huang, J. Zeng and Z. Cai, "Deep Reinforcement Learning-Based Traffic Signal," Entropy, 2019.

[18] Simulation of Urban Mobility, https://sumo.dlr.de/

[19] OpenAI Gym, https://gym.openai.com/

[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, 2020.

[21] B. R. Hajbabaie A., "Traffic Signal Timing Optimization Choosing the Objective Function," Journal of the Transportation Research Board, 2013.

[22] Keras deep learning API, https://keras.io/

[23] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," arXiv, 2015.