35th European Modeling & Simulation Symposium
20th International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2023 The Authors.
doi: 10.46354/i3m.2023.emss.040

Traffic Light Management for Automated Guided Vehicle Systems using Deep Reinforcement Learning

Ricardo Cardoso[1], Zafeiris Kokkinogenis[1,2,*], Rosaldo J. F. Rossetti[1,2] and João Emilio Almeida[2,3]

[1]Faculty of Engineering, University of Porto, Porto, Portugal
[2]Artificial Intelligence and Computer Science Laboratory (LIACC), Porto, Portugal
[3]Centro de Investigação e Tecnologias Avançadas, Instituto Superior de Tecnologias Avançadas, Porto, Portugal

[*]Corresponding author. Email address: kokkinogenis@fe.up.pt

Abstract

Intersection management in the presence of automated guided vehicles (AGVs) in industrial settings is a major problem that the artificial intelligence community tries to tackle. In this paper, we perform a comparative analysis of three well-known reinforcement learning techniques that produce policies to control a signalized intersection to coordinate traffic flow among automated guided vehicles. We implemented and tested four scenarios with different numbers of lanes and traffic light phase configurations. The analysis allowed us to gain critical insight into ways to improve the coordination of single intersections operating AGVs. The results suggest that decreasing the number of phases and increasing the number of lanes can be beneficial. As for the algorithms used, the Deep Q-Networks (DQN) and Double DQN performed better in simpler scenarios, whereas Dueling DQN seems to be more appropriate for more complex intersection settings.

Keywords: Traffic Light Control; Reinforcement Learning; Automated Guided Vehicles

1. Introduction

Nowadays, humans are required to drive vehicles reliably. However, if one can fully automate vehicle driving, it can provide a competitive advantage to the industries that use it. "Automated Guided Vehicles (AGVs) are mobile robots which are extensively used in the industry to transport goods from A to B" (De Ryck et al., 2020). They can be automobiles, trucks, drones, forklifts, or other industrial vehicles. Managing conflicts with AGVs is a complex task and several efforts in the Artificial Intelligence community try to address issues associated with their coordination. A prominent problem is the control of AGV traffic light intersections. The intersections can have different complexity levels; one possibility is to vary the number of roads that meet at each intersection or the number of lanes on each street. Combining more complex intersections, like roundabouts or multiple junctions, is possible as well. However, what they all have in common is that multiple vehicles might want to cross the same area simultaneously yielding countless conflicting situations. Therefore, it is desirable to develop a system capable of training models to handle any intersection, assuming that every vehicle on the road is autonomous (driven by a computer).

The hypothesis in this study is whether RL is a suitable methodology for representing control systems in dynamic environments such as AGV traffic light intersections. To assess such a hypothesis, we perform a comparative analysis of three well-known RL algorithms that model an intersection. We consider as the most suitable system the one capable of achieving the highest traffic flow while keeping collisions from happening.

The remainder of this paper is organized as follows. Section 2 presents the background knowledge and some relevant to support work herein reported. Section 3 discusses the methodological approach to designing intersection management strategies. In Section 4 we present and discuss the results of our experiments. Finally, Section 5 draws the conclusions and suggests routes for future work.

2. Background & Related Work

2.1. Q-Learning

According to Jang et al.(Jang et al., 2019) Q-Learning is a model-free RL method that allows agents to act optimally in Markovian domains, defined by a Markov Decision Process (MDP). Q-learning is an off-policy method because it separates the learning policy from the acting policy using Equation 1 to calculate the Q-values of the state-action pairs. The agent uses these values to select the best action to be performed at each state. In the same equation, $\alpha \in [0, 1]$ is the learning rate used by the agent in each iteration of the q-values update, R is the reward obtained by taking action a at state s, and γ is the reward discount factor. This process must be repeated several times until it converges to optimal values used to solve the problem at hand.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

2.2. Deep Q-Networks

With the evolution of Deep Learning, **Deep Q-Networks (DQNs)** emerged naturally. They can extract essential features from the data without human handcrafting domain-specific features (Sewak and Sewak, 2019). According to (Jang et al., 2019), DQNs combine Convolution Neural Networks (CNNs) with Q-learning. The key idea behind this method is to use experience replay to reduce the correlations between different states, actions, and rewards that the CNN could exploit. A replay buffer stores samples from the states, actions, and rewards and every time the CNN learns, random samples are extracted from the replay buffer to the CNN. However, remember that using more samples at each step means it takes longer for the CNN to update its values.

In combination with the experience replay technique, a target network ($Q_{\theta'}$) is used separately from the Q network (Q_θ) to increase training stability. The first one must be updated only periodically, while the second must be continuously updated. Thus, we end up with a new formula for the updates of the Q-values (Equation 2), where we use the target network $Q_{\theta'}$ to get subsequent Q-values for the next state-actions and the current network Q_θ to get the current Q-value for the current state-action.

$$Q_\theta(s,a) \leftarrow Q_\theta(s,a) + \alpha[R + \gamma \max_{a'} Q_{\theta'}(s',a') - Q_\theta(s,a)] \quad (2)$$

However, suppose the size of possible states S is considerable. In that case, before the agent can learn enough information from the environment, it may get stuck to exploiting the already explored environment, even if better options exist which have not yet been explored. A solution to this problem is to use **Double DQN (DDQN)** (Sewak and Sewak, 2019). This method still has a target $Q_{\theta'}$ and a current Q_θ network. The difference is that we select what action is the best to be taken from the Q network Q_θ, but we use the value from the target network ($Q_{\theta'}$) to update the Q network (Q_θ), as seen in Equation 3.

$$\begin{aligned} Q_\theta(s_t,a_t) \leftarrow & (1-\alpha)Q_\theta(s_t,a_t) + \\ & \alpha(R_{t+1} + \gamma Q_{\theta'}(s', arg\,max_{a'} Q_\theta(s',a')) \end{aligned} \quad (3)$$

Finally, **Dueling DQN** (Wang et al., 2016) branches the neural network into two sub-networks (Fig 1). The first one corresponds to the "**Value**" function $V(s)$ of each state, and the second one to the "**Advantage**" function $A(s,a)$ that computes the advantage of each action over the base value of being in state s. This separation does training much quicker: it has a global value for each state, updated upon any action. To get the Q-values for each state action, we must use Equation 4, where α is the parameter vector of the "Advantage" sub-network, β is the parameter vector of the "Value" sub-network, and θ is the parameter vector of the convolutional layer which is common to both networks (Sewak and Sewak, 2019).

$$\begin{aligned} Q_{(\theta,\alpha,\beta)}(s,a) = & V_{(\theta,\beta)}(s) + \\ & \left(A_{(\theta,\alpha)}(s,a) - \frac{1}{|A|}\sum_a A_{(\theta,\alpha)}(s,a)\right) \end{aligned} \quad (4)$$

The previous Equation (4) could be as simple as $Q(s,a) = V(s) + A(s,a)$; however, identificability would be lost. This means that the opposite would not be true despite being able to get the Q-value given the values of s and a. One could not retrieve the values of s and a from the given value of Q. The solution is to subtract the mean value of the advantage values for state s, given by $\frac{1}{|A|}\sum_a A_{(\theta,\alpha)}(s,a)$, from the advantage value of taking action a in state s.

2.3. Related Work

Intersection control in noisy scenarios where non-stationarity occurs not only due to the changing volume of vehicles, but also because of the random behavior in driving operational tasks is problem that research community constantly tries to tackle de Oliveira et al. (2006); Vilarinho et al. (2016).

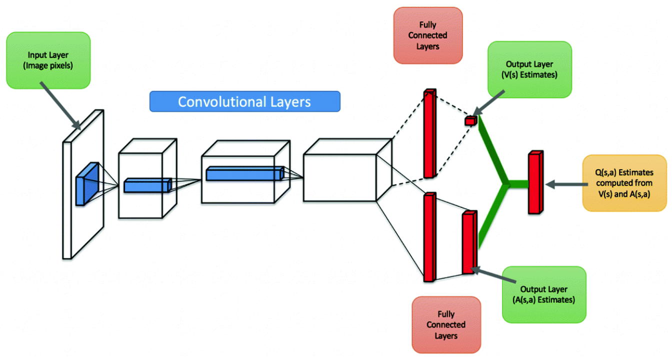

Input Layer (Image pixels)
Convolutional Layers
Fully Connected Layers
Output Layer (V(s) Estimates)
Q(s,a) Estimates computed from V(s) and A(s,a)
Output Layer (A(s,a) Estimates)
Fully Connected Layers

Figure 1. Schema of a Dueling Q-Network, from (Sewak and Sewak, 2019).

Authors in Cabrejas-Egea et al. (2021) considered a traffic signal control system using the leading RL approaches to date. Their goal was to compare those approaches to the ones used by commercial systems such as Fixed-Cycle Miller (1963) (seen in most traffic signals), MOVA Vincent and Peirce (1988), and SURTRAC Smith et al. (2013) in two scenarios: a)"Cross straight", where there is an intersection with four two-way one-lane roads where vehicles can only go straight through the intersection and b) "Cross triple", where there is an intersection with four two-way three-lane roads with one dedicated left turn lane. Their proposed solution uses three different methods: Double DQN (DDQN), Dueling Double DQN (DDDQN) and Advantage Actor Critic (A2C). The first two algorithms were used in combination with Prioritised Experience Replay that optimises the sample selection from the replay buffer by prioritising the temporal-difference error of the samples Wang et al. (2020). In Wei et al. (2018) is discussed the IntelliLight, "a Reinforcement Learning Approach for Intelligent Traffic Light Control". The proposed RL method to control traffic lights is tested it against real-world traffic data gathered from surveillance cameras and showed case studies of policies learned from that data. It defends that in real-world scenarios, the reward obtained by the RL agents is not enough to produce the best policy. Different policies can create the same reward. However one can be more suitable than the other if, for example, it has fewer phase changes.

The study in (Pálos and Huszák, 2020) conducts a comparison of Q-Learning based traffic light control methods and objective functions. It analyses the performance of DQN, DDQN, Dueling DQN and DDDQN methods to control traffic lights in a single intersection environment. The intersection is composed of four one-lane two-way roads where traffic can go straight through or turn right. Kavička et al. (2021) propose a study to assess traffic variants that use different traffic control signal plans, especially within a central intersection. In Chen et al. (2017) discusses an analytic hierarchy process method to determine the weights of evaluation indicators of traffic lights, together with microscopic simulation to optimize the signalized intersection for the objectives of efficient design and control. Both of these two efforts could be consider similar cases of controlling vehicles within a manufacturing compound or industrial warehouse. Authors in Pereira and Rossetti (2012) consider the integration of a microscopic traffic simulator with a game engine to simulated autonomous vehicles. Elbouzidi et al. (2022) presents a study that aims to assess the maturity of AI application within warehouse digital twins, namely techniques, objectives, and challenges. Finally, in Bruzzone et al. (2017), authors address the safety issues related to the development of new solutions based on autonomous systems for industrial applications.

3. Materials and Methods

This project aims to achieve Intelligent Traffic Light Management for AGV intersections (Fig 2). When replacing human drivers with computers controlling vehicles, new opportunities arise. There is no longer the need to use the typical yellow lights to let humans know they must slow down because the red light is coming soon. If the AGV receives a red signal (can be visual or through direct communication with the intersection controller), it will slow down; if it gets a green signal, it will proceed within its path. Another advantage of using AGVs is lower reaction times (humans need hundreds of milliseconds to start their reaction to a visual stimulus (Vis, 2006), compared to computer processing times in the nanoseconds range), so it is possible to change between light phases more often. Finally, computers are more predictable drivers than humans.

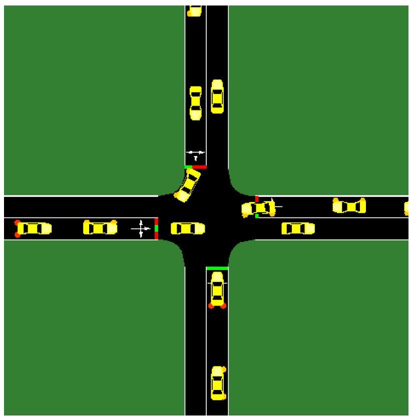


Figure 2. Example of an AGV intersection from SUMO (Lopez et al., 2018).

3.1. Problem Formalisation

The selected approach to solving the problem of Intelligent Traffic Light Management for AGVs are the DQN-based methods.

In the first place, DQN is an RL method, and therefore the problem must be defined as an MDP. To do so, the tuple $< S, A, P, R, \gamma >$ is defined by the following attributes:

- **Environment States (S)**
  - An RGB bird view image of the intersection.
  - A tensor with the speeds of the inbound and outbound vehicles from the intersection
  - Finally, four arrays, each one with information about the following features:
    * The percentage of space occupied for each lane.
    * The number of vehicles halting (with speed under 0,1 m/s) for each lane.
    * The sum of the vehicle's halting times for each lane.
    * The number of vehicles on each lane.
- **Action Space (A)** — For the action space, we use an integer value indicating the next phase of the traffic light. The phases are the different possible combinations for the green/red lights of the traffic signals and are predefined for each experiment.
- **Reward Function (R)** — The main goal of the reward function is to penalise collisions between vehicles while keeping a high traffic flow. To achieve the goal, six parameters define the reward function (Equation 5 and Table 1):
  - **S**: The average speed in m/s of all vehicles in oncoming lanes (that have not passed through the intersection yet). This is one of the most critical factors of the reward function because higher speeds directly lead to higher traffic flow.
  - **L**: It measures the throughput of the intersection. Higher throughput leads to more increased traffic flow.
  - **W**: The sum of waiting times from all vehicles in oncoming lanes. A vehicle is waiting if its speed is below $0.1 m/s$. It prevents the agent from allowing only one lane to pass through the intersection at high rates while keeping all the others at a standstill.
  - **I**: The number of vehicles in oncoming lanes. If many vehicles accumulate in oncoming lanes, the traffic signal must allow more vehicles to go through.
  - **P**: Traffic light phase factor (see Equation 6).
  - **C**: Collision factor (see Equation 7).

$$R = w_1 * S + w_2 * L + w_3 * W + w_4 * I + w_5 * P + w_6 * C \quad (5)$$

$$P = \begin{cases} 1, & last\_phase \neq current\_phase \\ 0, & last\_phase = last\_phase \end{cases} \quad (6)$$

$$C = \begin{cases} S, & collision = True \\ 0, & collision = False \end{cases} \quad (7)$$

Finally, the solutions are evaluated by letting the agents act in a traffic setup they have never seen before and measuring the obtained rewards, the number of accidents, and the throughput of the intersection during 3,600 steps (each step corresponds to one second of simulation, totalling one hour).

Table 1. Reward coefficients.

w_1	w_2	w_3	w_4	w_5	w_6
10	10	-1	-1	-10	-100

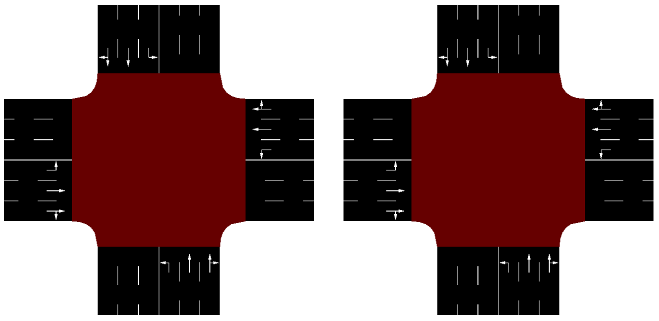

a) Intersection with four two-way one-lane roads. b) Intersection with four two-way three-lane roads.

Figure 3. Intersection

3.2. Methods and materials

The proposed solution to check whether or not RL is a suitable methodology to control AGV traffic light intersections consists in developing three different DQN-based agents, using **DQN**, **Double DQN**, and **Dueling DQN**. Those agents are trained for 100,000 steps in 6 different scenarios (Subsection 3.3), following the Training Pipeline Loop (Subsection 3.4). Afterwards, a final testing loop of 3,600 steps is performed to evaluate the trained agents (Subsection 3.5).

3.3. Scenarios

A total of six different scenarios were developed to train and test each agent. All the scenarios consist of an intersection with four two-way roads controlled by a traffic light, where only AGVs with the same characteristics are allowed to go through. The differences between the scenarios are divided into two groups:

- **Number of lanes**: Regarding the number of lanes on the roads, there are two possibilities (Figs 3a and 3b). In Fig 3a, there is an intersection where all the roads have **one lane** in each direction. That one lane is responsible for traffic going in all three possible directions (left, front, and right). Fig 3b shows an intersection where all roads have **three lanes** in each direction. The left lane handles traffic turning left, the middle lane only handles traffic going straight, and the right lane handles traffic going straight and turning right.
- **Number of phases**: There are three different phase setups for each intersection in the previous point (Figs

4 and 5). In Fig 4, there is a setup with **2 phases**. The first one allows traffic to flow from north and south directions, while the second allows traffic to flow from east and west directions. In this setup there is actually a third phase that blocks traffic from all directions to let the agent clear the intersection if it wants to. Fig 5 shows a traffic light setup with **4 phases**. Going from (a) to (d), each phase allows traffic to flow from north, east, south, or west, respectively. As seen in the 2 phases setup, the 4 phases setup also has a fifth phase that blocks traffic from all directions.

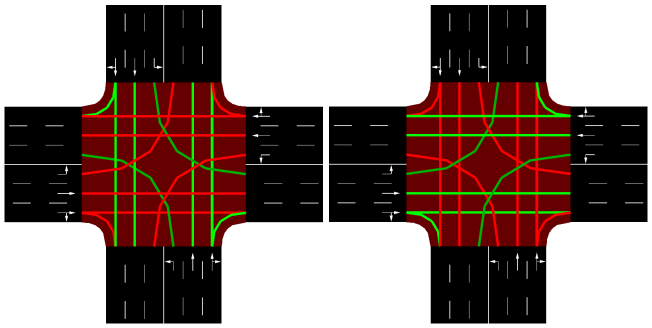


Figure 4. Traffic light with 2 phases.

3.4. Training Pipeline Loop

Before the DQN Agent gets into the Training Pipeline Loop (Fig 6), it receives an initial representation of the starting state of the simulation from the Environment module, and then the loop starts (steps 1 to 4).

- **Step 1**: The Agent tells the Environment which action it chooses (an action corresponds to setting a phase at the traffic light, as seen in Section 3.1). Selecting an action depends on some randomness defined by the coefficient between exploration and exploitation (ϵ). defined as an exponential function (Eq. 8) to decrease that value until a minimum of 0.10 during training.

$$\epsilon(s) = \begin{cases} 0.5 * 0.999^{s}, & 0.5 * 0.999^{s} \geq 0.1 \\ 0.1, & \textit{otherwise} \end{cases} \tag{8}$$

- **Step 2**: The Environment, directly connected to SUMO via TraCI (Lopez et al., 2018), performs the chosen action. Upon exchanging information with SUMO, the Environment builds the new state representation, detailed in the previous section, calculates the value/reward of that state, and sends that information (state and reward) back to the Agent.
- **Step 3**: The tuple (Previous State, Action, Reward, State) is pushed to a Replay Buffer that stores up to 5,000 tuples, acting as the Agent's memory.
- **Step 4**: A batch of 64 random tuples is sampled from the Replay Buffer to train the Agent's neural network. This step is only processed after 64 steps of simulation to have enough tuples to perform the batch training of the Agent. Despite only performing 100,000 steps of simulation, the use of the Replay Buffer to train the agent allows it to repeat past experiences and update the neural network almost 6,400,000 times.

3.5. Testing Pipeline Loop

The Testing Pipeline Loop is based on the training one but without the training features. Only one hour of simulation is performed (3,600 steps). All actions are chosen using the Agent's policy (no more exploration is needed), and no updates are done in the Agent's neural network, so there is no need to have the Replay Buffer. This loop aims to collect data about the performance of the multiple agents in different scenarios.

3.6. Simulating the Intersection

To simulate the intersection we use the SUMO traffic simulator (Lopez et al., 2018). It is possible to interact with the graphic interface of the simulator through TraCI, creating a *view* centred in the middle of the intersection and recording the frames of the *view* at every step. The video frames are then used to build the state representation. Through TraCI we can collect further data to augment the environment state with additional information. For the purpose of this paper we have devised two scenarios, one for the one-lane and another for the three-lane configurations. They both consist of four perpendicular roads with 100 meters that meet at a traffic light controlled intersection. We have created two route files for each scenario, one for training and another for testing. The training files contain *flows* (a *flow* repeatedly generates vehicles in the same path with different departure times) from every road to every other road. According to it, our scenarios have four roads and 12 *flows*, three from each road. In each *flow*, we defined a probability of 0.05 for a new vehicle joining the simulation in each step. The testing files were generated with a different method to remove randomness and compare all agents with the same traffic demands.

4. Results and Discussion

4.1. One Lane Scenario

4.1.1. 2 Phases

Here we are analysing the scenario where we have one-lane roads and a two-phased traffic light. The most obvious conclusion we can take by generally looking at the graphs from Fig 8 is that the agent with the worst performance during training is Dueling DQN. It shows the lowest values for rewards and average speeds and the highest values for average waiting times and the number of collisions. However, despite having the worst results, Dueling DQN shows to be the most stable approach. Comparing DQN with Double DQN, we see that DQN can achieve marginally

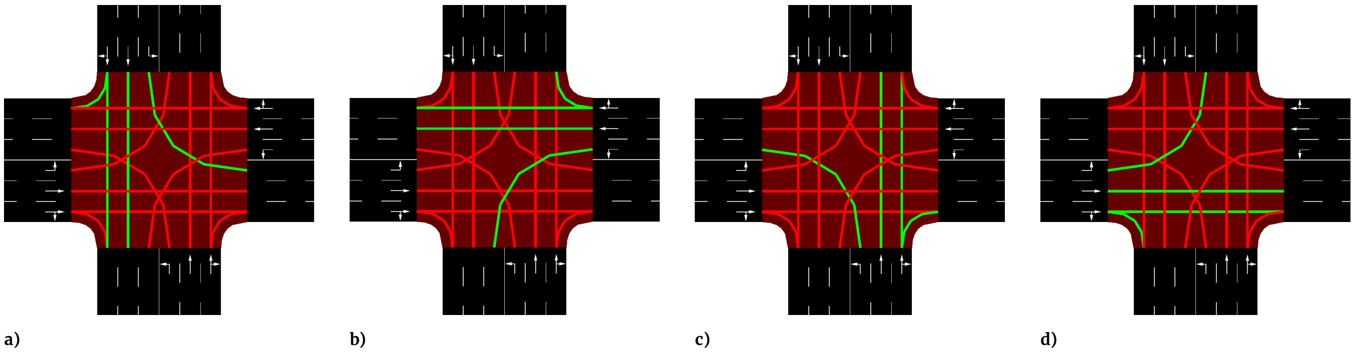


Figure 5. Traffic light with 4 phases.

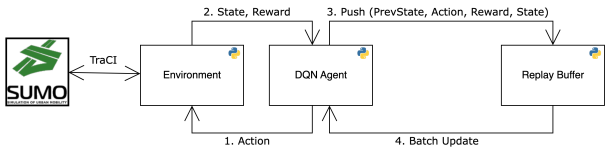


Figure 6. Training pipeline for DQN-based agents.

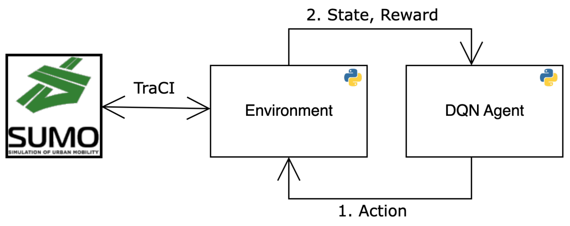


Figure 7. Testing pipeline for DQN-based agents.

better results. However, the results from DQN and Double DQN are not stable. This shows that these agents might be over-fitting to some experiences, and whenever it faces a new situation, the agents do not know what the best action is.

4.1.2. *4 Phases*

Now, analysing the scenario where we have roads with one lane and the traffic light has four phases (Fig 9), the Dueling DQN agent gets the worst overall results once again. However, it still manages to be the more stable agent in every metric. The main contributing factor to being more stable is using two separate networks (Value and Advantage, seen in Section 2.2) to obtain the Q-values. One of the networks has the value of each state, and the second has the advantage of taking one action over the others in that state. Even if the agent faces a state-action pair it has never faced before, it might have faced that same state with another action. So, despite no value being stored in the Advantage network for that action, there will be a stored value in the Value network for that state, stabilising the training. If the DQN and Double DQN agents have never faced that state-action pair, they will have no idea of its value and will be in an entirely new situation. This makes training less stable while giving better results for known situations (known as over-fitting to known situations).

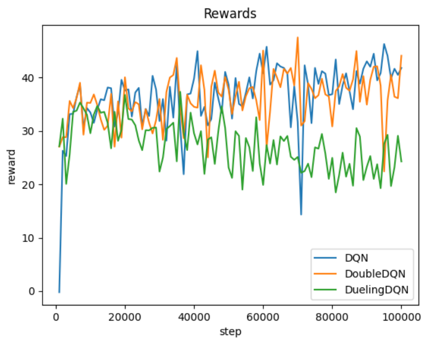
4.2. Three Lanes Scenario

4.2.1. *2 Phases*

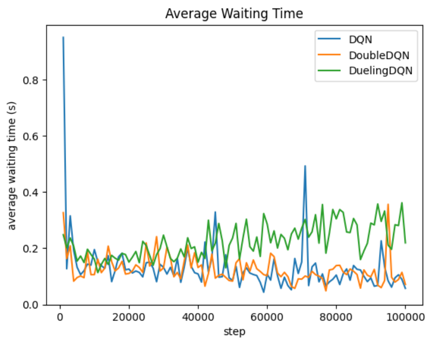
Looking at the results from the scenario with three lanes and two phases (Fig 10), it is possible to see that they follow the same tendency as the ones from the scenario with one lane and two phases (Fig 8, page 7), except in the collisions graph. The absence of collisions is due to the way SUMO's vehicles handle this intersection and not due to agents' actions. Once more, Dueling DQN was the worst performer agent. Also, there were more stable agents this time. Having an action space with only three possibilities (the two phases plus all lights red) puts most of the responsibility on the agent's Value network. Adding two lanes on each road makes the observation space increase considerably, destabilising the Value network. Comparing DQN with Double DQN, this time, DQN appears to be more stable, which is expected as we have a low number of possible actions. The advantages of Double DQN over DQN are best noticed in problems with large action spaces.

4.2.2. *4 Phases*

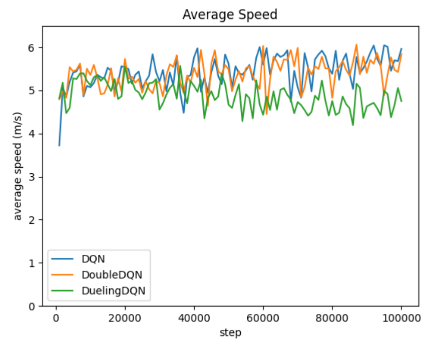
When we have the same three-lane roads, but with four-phase traffic lights, the training of all agents stabilises (Fig 11). As in the previous scenario, Dueling DQN is the worst performer and the less stable agent. The differences between DQN and Double DQN are negligible, and they are expected to be the top performers in this scenario. Comparing this scenario with the one-lane scenario from Fig 9 (page 7), all agents managed to be more stable and achieve better results. Fig 5 from page 6 shows that only traffic from one road can move at a time. In the three-lane scenario, this means that three vehicles can pass through the intersection simultaneously (one on each lane), while in the one-lane scenario, only one vehicle can go through at each time. The result of those differences is that the three-lane scenario can achieve higher speeds, lower waiting limes, and, consequently, higher rewards, no matter what action they choose.



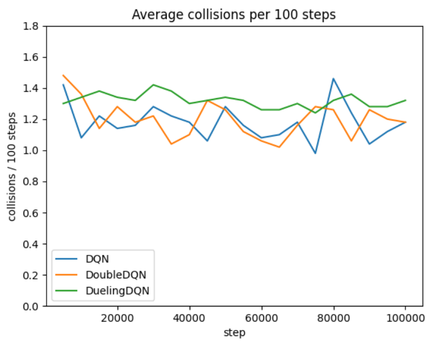
a) rewards



b) average waiting times

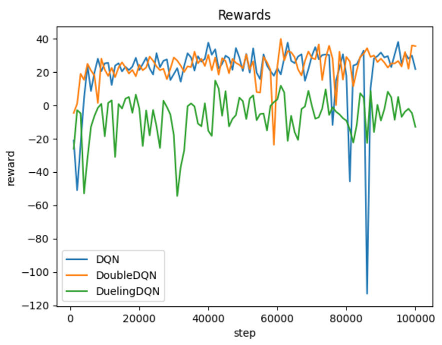


c) average speeds

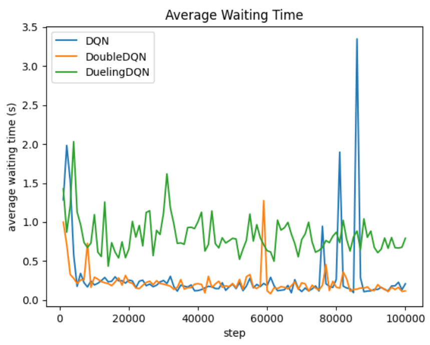


d) and average number of collisions

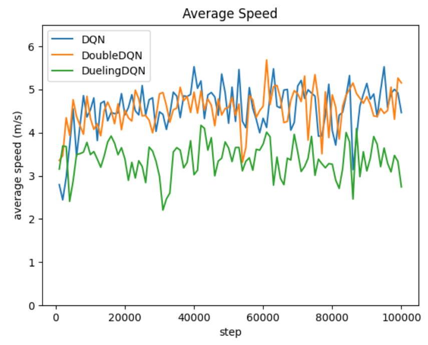
Figure 8. Graphs for rewards during training in scenarios with one-lane roads and two phases.



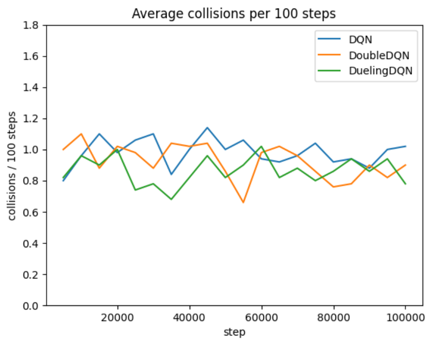
a) rewards



b) average waiting times



c) average speeds



d) and average number of collisions

Figure 9. Graphs during training in scenarios with one-lane roads and four phases.

4.3. Testing

We tested the trained agents in the same traffic demand for each scenario, as explained in Subsection 3.6 on page 5. This means that the same number of vehicles enter the simulation at the same time steps and follow the same routes for all the agents. During testing, we analyse three metrics: the average rewards, the total number of accidents, and the throughput (number of vehicles that passed through the intersection) of each agent for 3,600 steps (equivalent to one hour).

4.4. One Lane

DQN is a good agent in scenarios with one-lane roads. However, if we see the subsection 4.1, it was the least stable agent and suffered the most from over-fitting. That becomes clear in the four-phase scenario from Table 2, where it obtained a highly negative reward despite having a low number of accidents and a high throughput. In the 4,096 testing simulation, it is clear that the DQN agent exploited one action where only cars from one road can go through, keeping high speeds for that road and high throughput for the overall system. However, all the other roads stayed at a standstill for most of the simulation, lowering the reward value. This is a methodology to avoid in a real-life scenario due to its stability and over-fitting issues.

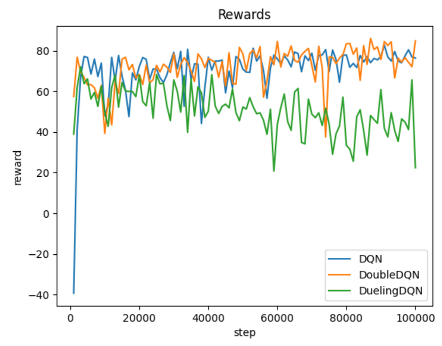
Table 2. Performance of DQN, Double DQN, and Dueling DQN agents in scenarios with one-lane roads.

Nr. Phases	Metric	DQN	Double DQN	Dueling DQN
2	average reward	9	-2690	**10**
	number of accidents	51	**37**	50
	throughput	**1533**	1497	1422
4	average reward	-1347	**8**	-6
	number of accidents	**18**	47	43
	throughput	**1550**	1493	1371

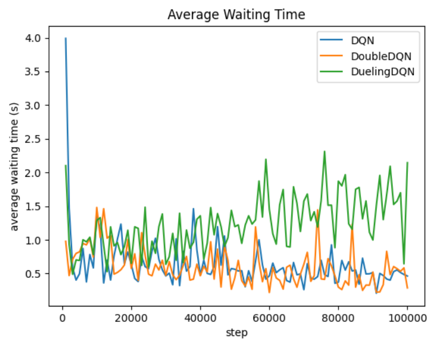
Comparing Double DQN with Dueling DQN, selecting which one is better is not straightforward. Dueling DQN is more consistent than Double DQN. However, it tends to generate more collisions, which can be a problem related to the reward function used and not the method itself. The Double DQN still manages to be a good choice, especially in scenarios with large action spaces.

4.5. Three Lanes

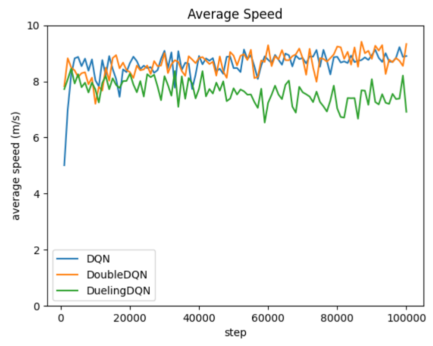
In the three-lane scenarios, the results were different (Table 3). All agents managed to keep the intersection collision-free with similar through-puts in scenarios with two or four phases, which is excellent. The rewards of the Dueling DQN are lower than the other agents due to the speeds and waiting times of the vehicles. Both DQN and Double DQN agents managed to keep the intersection free of long queues of vehicles (consistently below five vehicles), while the Dueling DQN agent occasionally formed a



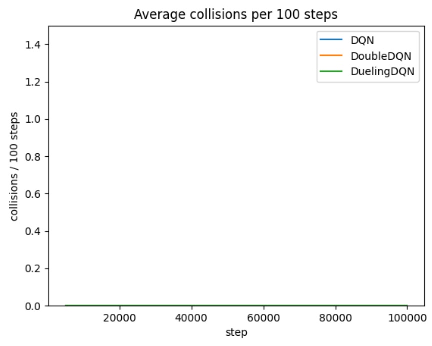
a) rewards



b) average waiting times

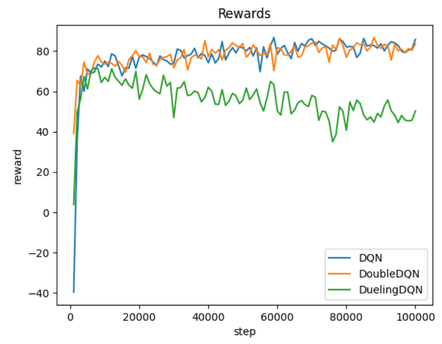


c) average speeds

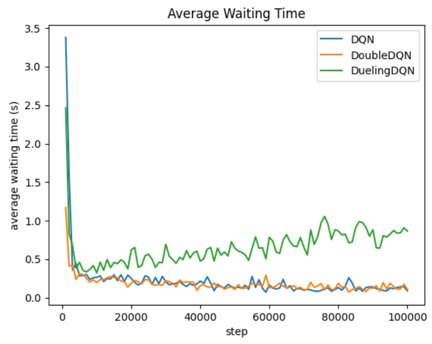


d) and average number of collisions

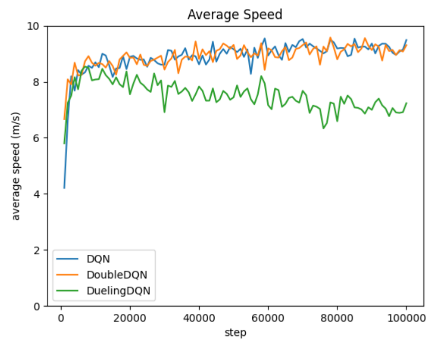
Figure 10. Graphs during training in scenarios with three-lane roads and two phases.



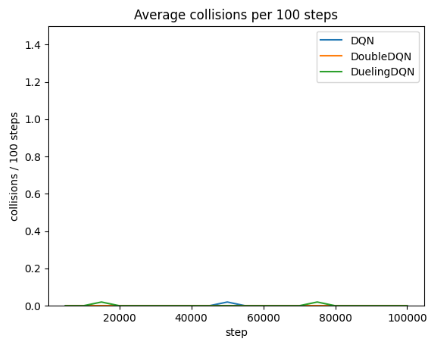
a) rewards



b) average waiting times



c) average speeds



d) and average number of collisions

Figure 11. Graphs during training in scenarios with three-lane roads and four phases.

Table 3. Performance of DQN, Double DQN, and Dueling DQN agents in scenarios with three-lane roads.

Nr. Phases	Metric	DQN	Double DQN	Dueling DQN
2	average reward	52	**53**	-81
	number of accidents	0	0	0
	throughput	**2700**	2699	2682
4	average reward	**67**	**67**	1
	number of accidents	0	0	0
	throughput	**2699**	2697	2691

big queue on one of the roads, but only for short periods.

Finally, Table 3 shows that increasing the number of phases in a traffic light is not adequate to keep traffic flowing.

5. Conclusions

In this paper we consider the problem of controlling AGVs in signalized intersections. Such scenarios are typical in different industrial settings, such as warehouse shop-floors, where the crossing of AGVs needs to be coordinated. We formalised the problem of Intelligent Traffic Light Management as a MDP. The states of environment have information about the number of vehicles on each lane, the waiting times, the number of waiting vehicles, the percentage of space occupied, the speeds of each vehicle, and a video frame of the intersection. The action space consists of the possible traffic light phases. We performed a preliminary analysis and evaluated the performance of three well-known RL methods: DQN, Double DQN and Dueling DQN. We conclude that for the task at hand the DQN approach shows to be the least stable one and suffers from over-fitting to the initial observations of the training. The Double DQN method shows a fair exploration of the action space, leading to more stable and better results. The Dueling method agent does not have the best results, however, it is highly stable, especially when the action space increases. Future work will consider more complex scenarios with multiple intersections and a more extensive analysis of other RL approaches. Furthermore, a step forward in improving this work is to explore Multi-Agent Reinforcement Learning (MARL) methods in controlling multiple AGV intersections. However, these methods bring about new challenges, similar to those that Nguyen et al. explore in Nguyen et al. (2020), such as non-stationarity, partial observability issues, transfer learning in multi-agent reinforcement learning techniques, and training schemes among others.

6. Acknowledgment

This work is a result of project DynamiCITY: Fostering Dynamic Adaptation of Smart Cities to Cope with Crises and Disruptions [reference NORTE-01-0145-FEDER-000073] supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF). It was also financially supported by Base Funding - UIDB/00027/2020 of the Artificial Intelligence and Computer Science Laboratory – LIACC - funded by national funds through FCT/MCTES (PIDDAC).

References

Bruzzone, A. G., Massei, M., Agresta, M., Di Matteo, R., Sinelshchikov, K., Longo, F., Nicoletti, L., Di Donato, L., Tomassini, L., Console, C., Ferraro, A., Pirozzi, M., Puri, D., Vita, L., Cassandra, F., Mennuti, C., Augugliaro, G., Site, C. D., Di Palo, F., and Bragatto, P. (2017). Autonomous systems & safety issues: The roadmap to enable new advances in industrial applications. In *29th European Modeling and Simulation Symposium, EMSS 2017, Held at the International Multidisciplinary Modeling and Simulation Multiconference, I3M 2017*, pages 565–571.

Cabrejas-Egea, A., Zhang, R., and Walton, N. (2021). Reinforcement learning for traffic signal control: Comparison with commercial systems. *Transportation research procedia*, 58:638–645.

Chen, Y., Li, L., Shen, Y., and Liu, B. (2017). A simulation optimization method for design and control of signalized intersections in urban areas. In *29th European Modeling and Simulation Symposium, EMSS 2017, Held at the International Multidisciplinary Modeling and Simulation Multiconference, I3M 2017*, pages 1–7.

de Oliveira, D., Bazzan, A. L., da Silva, B. C., Basso, E. W., Nunes, L., Rossetti, R., de Oliveira, E., da Silva, R., and Lamb, L. (2006). Reinforcement learning based control of traffic lights in non-stationary environments: A case study in a microscopic simulator. In *CEUR Workshop Proceedings. 4th European Workshop on Multi-Agent Systems (EUMAS'06)*, pages 31–42.

De Ryck, M., Versteyhe, M., and Debrouwere, F. (2020). Automated guided vehicle systems, state-of-the-art control algorithms and techniques. *Journal of Manufacturing Systems*, 54:152–173.

Elbouzidi, A. D., Bélanger, M. J., El Cadi, A. A., Pellerin, R., Lamouri, S., Valencia, E. T., and Lamouri, S. (2022). The role of ai in warehouse digital twins. In *34th European Modeling & Simulation Symposium (EMSS'2022)*, pages 172–179.

Jang, B., Kim, M., Harerimana, G., and Kim, J. W. (2019). Q-learning algorithms: A comprehensive classification and applications. *IEEE access*, 7:133653–133667.

Kavička, A., Diviš, R., Bažant, M., and Křivka, P. (2021). Simulations of road traffic at light-controlled intersections. In *Proceedings of the 33rd European Modeling & Simulation Symposium (EMSS 2021)*, pages 33–38.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wiessner, E. (2018). Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582.

Miller, A. J. (1963). Settings for fixed-cycle traffic signals. *Journal of the Operational Research Society*, 14(4):373–386.

Nguyen, T. T., Nguyen, N. D., and Nahavandi, S. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 50(9):3826–3839.

Pálos, P. and Huszák, Á. (2020). Comparison of q-learning based traffic light control methods and objective functions. In *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6. IEEE.

Pereira, J. L. and Rossetti, R. J. F. (2012). An integrated architecture for autonomous vehicles simulation. In *Proceedings of the 27th annual ACM symposium on applied computing*, pages 286–292.

Sewak, M. and Sewak, M. (2019). Deep q network (dqn), double dqn, and dueling dqn: A step towards general artificial intelligence. *Deep Reinforcement Learning: Frontiers of Artificial Intelligence*, pages 95–108.

Smith, S., Barlow, G., Xie, X.-F., and Rubinstein, Z. (2013). Smart urban signal networks: Initial application of the surtrac adaptive traffic signal control system. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 23, pages 434–442.

Vilarinho, C., Tavares, J. P., and Rossetti, R. J. F. (2016). Design of a multiagent system for real-time traffic control. *IEEE Intelligent Systems*, 31(4):68–80.

Vincent, R. A. and Peirce, J. R. (1988). 'mova': Traffic responsive, self-optimising signal control for isolated intersections. *TRRL RESEARCH REPORT*.

Vis, I. F. (2006). Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3):677–709.

Wang, X., Xiang, H., Cheng, Y., and Yu, Q. (2020). Prioritised experience replay based on sample optimisation. *The Journal of Engineering*, 2020(13):298–302.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR.

Wei, H., Zheng, G., Yao, H., and Li, Z. (2018). Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2496–2505, New York, NY, USA. Association for Computing Machinery.