



# Reinforcement learning for traffic light control with emphasis on emergency vehicles

Mahboubeh Shamsi<sup>1</sup> · Abdolreza Rasouli Kenari<sup>1</sup>  · Roghayeh Aghamohammadi<sup>1</sup>

Accepted: 3 September 2021 / Published online: 14 September 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

## Abstract

Traffic lights are an important controlling factor in traffic flows, and good policies will facilitate traffic congestion. A car's waiting time is highly related to the period in which the traffic lights are green or red; thus, a proper controlling policy reduces the waiting time and speeds up the car movement. Emergency vehicles always have a higher priority than other cars and need to reach their destination faster. In the last decade, a lot of research has been done to solve the traffic light scheduling issue using artificial intelligence techniques. Reinforcement learning with simulating human learning has the outstanding performance to solve complex problems, such as traffic lights control. In this study, a deep reinforcement learning traffic light control was implemented at a crossroad using real-time traffic information with an emphasis on emergency vehicles. The agent learns to adjust its policies to prioritize the emergency cars over the other cars. The results show less waiting time and speed up the cars' movement. It also indicates the emergency vehicles will cross the intersection with the least delay. With 200 to 600 cars and one emergency vehicle, the average delay decreases 2–16% in total using reinforcement learning. Considering more emergency vehicles, there is a 27–40% decrease in average delay for emergency vehicles.

**Keywords** Urban traffic control · Traffic light · Reinforcement learning · Emergency vehicles · Delay reduction

---

✉ Abdolreza Rasouli Kenari  
rasouli@qut.ac.ir

Mahboubeh Shamsi  
shamsi@qut.ac.ir

Roghayeh Aghamohammadi  
Aghamohmmadi@qut.ac.ir

<sup>1</sup> Faculty of Electrical and Computer Engineering, Qom University of Technology, Qom, Iran

## 1 Introduction

Today, the number of vehicles is increasing exponentially, but the capacity of the roads and transport systems has not been adequately developed to cope with vehicle traffic. Traffic conditions are complicated and turbulent. According to a report [1], large amounts of time and money are wasted, for example, in US urban areas, traffic congestion between years 2000 and 2010 causes 5.5 billion hours of delay and 2.9 billion gallons of fuel waste. It is predicted which congestion costs will increase from \$121 billion (in 2011) to \$199 billion (in 2020). Traffic accidents occur due to malfunctions in traffic lights and driver's negligence. Improved performance in traffic management systems increases the safety and efficiency of transportation systems. An old traffic control system used static times for intersection management and did not prioritize emergency vehicles such as ambulances, firefighters, and police cars. It caused the loss of life, damage or loss of property, and increased fuel costs, pollution, and congestion. An intelligent traffic management system achieves efficient traffic management during emergency events using communication technologies and processing technologies and smart system algorithms [2]. In summary, traffic congestion causes air pollution, wastes resources and time, reducing the quality of life, and endangers human life. Emergency vehicles, meanwhile, need more management to reach their destination quickly and uninterruptedly.

Optimization of traffic control systems is at the heart of traffic management operations. Many solutions for designing an efficient traffic signal pattern rely on controllers using pre-scheduling steps. Such systems are not capable of detecting dynamic changes in local traffic flow and therefore cannot adapt to new traffic conditions. A new and alternative approach has been proposed by computer scientists to design a traffic light controller that relies on the use of intelligent agents. The idea is to allow autonomous beings, agents, to learn optimal behavior by interacting directly with the system. By using machine learning algorithms based on the reward assignment based on the results of agent-selected actions, a control policy is attempted to optimize the traffic flow [3].

This research on intelligent transportation systems (ITS) focuses on traffic management at a crossroads. Deep reinforcement learning is used to control the traffic light. First, the agent, who is a traffic light, learns from the traffic flow. Due to the distribution of vehicles at the intersection and duration of the car stops, experience and learning take place. The driving time of the traffic light is adjusted dynamically according to the distance of the last car within the respective lane. Emergency vehicles are also being considered, and efforts have been made to prioritize and minimize the delay. The traffic status of the lanes is constantly monitored and if the waiting time of the emergency vehicle is reduced, it will minimize the impact on public lane traffic.

The main contributions of the research are as follows:

- The formal model of crossroad traffic, such as its state space, action space, and reward function, is defined with an emphasis on emergency vehicles.

- An intelligent traffic light agent using deep reinforcement learning is implemented.
- The average delay and waiting time of the cars who stopped before the traffic light is significantly reduced.
- The traffic light agent adjusts his policies to prioritize the emergency vehicle over other vehicles.

The rest of this paper is organized as follows: In Sect. 2, some of the previous related researches are investigated. In Sect. 3, reinforcement learning and reinforcement learning algorithm in traffic control are described. In Sect. 4, the setting of implementation and results are presented, and in Sect. 5, the conclusion is deduced.

## 2 Related work

The traffic control problem is a nonlinear time-variable optimization control problem, and the dynamic parameters of its structure are difficult to identify, so it is difficult to define a traditional mathematical model. Artificial intelligence technology has another way of dealing with such problems [4]. Intelligent methods for traffic control can be divided into several categories: fuzzy logic, neural networks, evolutionary algorithms, and learning agents [4].

Fuzzy logic is a powerful tool for processing nondeterministic and nonlinear problems [4]. In [5], Alam et al. have proposed a fuzzy logic-based traffic light control system, according to the number of vehicles sensed at the fuzzy controller input. Queue length is defined as very short, short, medium, large, and very large fuzzy variables. Green phase extension is determined as a fuzzy variable including zero, short, medium, and large. The simulation results show that the fuzzy controller performs better than the fixed time controller and is more cost-effective. Fuzzy optimization is also more flexible than traditional techniques.

Homaei et al. [6] have proposed a fuzzy intersection controller concerning emergency vehicles, that is, controlling the intersection traffic alongside the preemption of traffic in front of the emergency vehicle and deploying the proposed controller. This research proposes two main functions: phase selector fuzzy traffic light and fuzzy green phase extender. The first function identifies the next green phase, and the controller of the green phase extension decides to develop or terminate the current green phase. Finally, the results are compared with the pre-scheduling control system, and the performance of this method is shown. But there are still drawbacks in fuzzy logic, the configuration of techniques and settings are experimental, and there is no theory that can demonstrate the robustness of methods [7].

Genetic algorithm is one of the most well-known evolutionary algorithms that is widely used in urban traffic problems. The goal of the genetic algorithm is to obtain an approximate solution to an optimization problem where there are no exact methods to solve it in a reasonable time. This algorithm approaches the solution with successive mutations [7].

Singh et al. [8] proposed a real-time traffic control method using the genetic algorithm to provide relatively optimal performance for intersections. This

intelligent system makes real-time decisions on green time extensions that ultimately determine the optimal green time extension, where, and how long green time. In the end, it compared the experimental results with the time constant method and showed that a great improvement has been achieved.

Park et al. [9] developed a genetic algorithm for optimizing traffic time control, which can also control intersections in saturated conditions. The genetic algorithm is used to search for a relatively optimal schedule. It is compared with a program included in the simulator and investigated in clear, moderate, and crowded conditions, which has shown improved results in clear and crowded conditions.

However, evolutionary algorithms such as genetics, when faced with large-scale problems, take too much time to converge optimally. Therefore, the use of evolutionary algorithms for traffic control is limited [4]. Besides, this method requires a lot of computation, its parameters are difficult to tune, and it can cause delays to reach the results [7].

The artificial neural network is widely used in a variety of areas such as automatic control, pattern recognition, and information and signal processing; these applications are due to capabilities such as generalization power, self-adaptation, self-learning, and self-management [4].

Febbraro et al. [10] have used a neural network to solve the traffic control problem that provides traffic variables as inputs and the network is capable of extracting effective features and does not need to identify parameters during changes. In other words, the most important advantage is that you do not need to know the dynamic characteristics of the traffic. It is obvious that the more accurate the prediction, the better control, and optimization strategy. A comparison between a neural network and a classical model is performed and the results are shown.

Dhingra et al. [11] used traffic classification to determine traffic control and management strategies. By classifying traffic and discovering congested lanes, one can examine the causes and consequently make timely decisions to modulate lane traffic congestion. Surveillance video data have been used as a good source for categorizing lane traffic using a convolutional neural network, which requires at least some pre-processing compared to other categorization algorithms. The video is categorized by its traffic rate. The convolutional neural network is first trained and then used for evaluation and testing.

One of the main disadvantages of the artificial neural network is that it is difficult to interpret and determine its parameters [7]. The learning process must converge at the optimal global point. Further development and enhancement of artificial neural networks are needed to use this approach in traffic control [4]. A summary of the reviewed researches is presented in Table 1.

Unlike supervised or unsupervised learning, reinforcement learning teaches optimal policy by understanding the states of the environment and receiving nondeterministic information from the environment. This method of learning is a trial-and-error process. The agent is only able to obtain the value of the action performed at present. The agent must optimize his policy by using state information, state change, actions, and rewards [4]. One of the benefits of reinforcement learning is that there is no need for a mathematical model to describe the external environment [4].

**Table 1** Summary of researches

Method	Researches	Method function	Advantages	Challenges
Fuzzy logic	Alam et al. (2013) [5] Homaei et al. (2015) [6]	Extension of the green phase Extension and selection of green phase	Processing the non-deterministic and nonlinear problems	The configuration and set up is experimental
Genetic algorithm	Singh et al. (2009) [8] Park et al. (1999) [9]	Extension of the green phase Optimized scheduling program	Finding an approximate solution to an optimization problem in a reasonable time with sequential mutations	Long time to converge on big issues, high computation, and difficult parameter setting
Neural network	Febbraro et al. (2005) [10] Dhingra et al. (2019) [11]	Traffic prediction and control Traffic classification and control	Power of generalization, self-adaptation, self-learning, self-management	Hard determination and interpretation of parameters, learning only according to available data
Reinforcement learning	Vidali et al. (2019) [12] Mousavi et al. (2017) [13] Gao et al. (2017) [14] Genders and Razavi (2016) [15] Li et al. (2016) [16] Chu et al. (2019) [26]	Deep Q-learning with a fully connected multilayer neural network Value-based reinforcement learning and policy-based approaches Hypothetical partition of the road plus the relative speed of the vehicles—CNN Spatial representation of the environment—CNN Deep stack autoencoder neural network to learn Q-values A modified Q-learning mechanism	The absence of a training dataset, Similar to the learning of human beings, correct the errors that occurred during the training process, maintain a balance between exploration and exploitation, solve very complex problems without mathematical complex theory	Needs a lot of data and a lot of computation, overload of states,

Table 1 (continued)

Method	Researches	Method function	Advantages	Challenges
	Kumar et al. (2019) [27]	Efficient dynamic traffic light control system using deep reinforcement learning		
	Wei et al. (2018) [28]	IntelliLight, reinforcement learning focusing on traffic policies and generate traffic rules		
	Bouktif et al. (2021) [29]	Hybrid action space deep reinforcement learning		
	Maske et al. (2020) [30]	scalable, decentralized deep reinforcement learning		

One of the most common methods used in reinforced learning for traffic control is the combination of the Q-learning algorithm and deep neural network, which yields optimal operating behavior with neural networks used to estimate Q-values of a given state [12]. Some research approaches use Q-learning and convolutional neural networks to calculate environment state and feature learning from an image [13] or a spatial representation [12, 14, 15].

Genders and Razavi [15] and Gao et al. [14] used a convolutional neural network to learn features from the spatial representation of the environment, followed by a fully connected network to generate outputs that are Q-values. The method [14] uses the average volume of information in each cell of the cell derived from the hypothetical partition of the road plus the relative speed of the vehicles as the state space. Method [15] considers the current phase next to the road cell vector. It has shown better results than the first longest and time-constant policies. The method [10] is compared with the shallow neural network method, which has shown good results.

Mousavi et al. in the study [13] also used the CNN network to derive the feature from the image and estimate the Q-value calculation function. In this study, two methods of traffic control are analyzed: value-based reinforcement learning and policy-based approaches. In the first method, the value of the actions is predicted by minimizing the mean squared error of the Q-values with the gradient descending method. Alternatively, the policy learned will be improved by updating policy parameters to increase the likelihood of doing good. The output of the value-based method is the value of the actions, and the output of the policy-based method is the probabilistic distribution of actions. The results of both methods have been tested and are better than the single-layer shallow neural network. Due to the use of images of the environment as a state, the state space of this research is very large and massive.

Li et al. [16] used a deep stack autoencoder neural network to learn Q-values to minimize the error between Q-value prediction by the neural network and the target Q-value by assigning a loss function. The results are compared with the conventional reinforcement learning method and shown to be better. Vidali et al. [12] have used the deep Q-learning method to control traffic lights; two scales have been reviewed and compared for reward measures. The proposed scale, which calculates the total waiting time, has shown better results. This method uses a fully connected multilayer neural network. The proposed state space is a vector of the environment division.

Chu et al. [26] adjusted the traffic lights cycle dynamically with traffic data to reduce degrees of traffic congestion. A modified Q-learning mechanism has been applied to search for traffic lights cycle configuration with optimal delay time, and less processing time. Kumar et al. [27] proposed an efficient dynamic traffic light control system using deep reinforcement learning. They capture real-time traffic information from installed traffic sensors. Their deep reinforcement learning model with vehicle heterogeneity changes the traffic signal phase and duration dynamically. Results are compared with fixed-time model, and dynamic traffic control and show less average waiting time.

Wei et al. [28] designed IntelliLight, an intelligent traffic light system using reinforcement learning to replace traditional traffic lights' hand-crafted rules. The

advantage of their work is focusing on traffic policies, and the system can learn policies from the real data. The system also can help generate traffic rules for real-world applications. Bouktif et al. [29] used hybrid action space deep reinforcement learning for traffic signal control. They focus on the combination of both discrete and continuous approaches. In discrete control, the agent selects the traffic light phase from a finite set of phases, whereas in the continuous control approach, the agent decides within a sequence of phases. The advantage is a hierarchical decision-making process based on a Parameterized Deep Q-Networks (P-DQN) Architecture. They decrease the average queue length of vehicles and the average travel time. But, the IntelliLight system and P-DQN are implemented for one simple intersection and should be revised for real-world multi-intersection problems.

Maske et al. [30] introduced a scalable, decentralized deep reinforcement learning (DRL) scheme for controlling traffic signalization. They solved the problem of the multi-intersections with a multi-agent DRL, with a new state representation and reward definitions. They minimized the average time and improved the learning process.

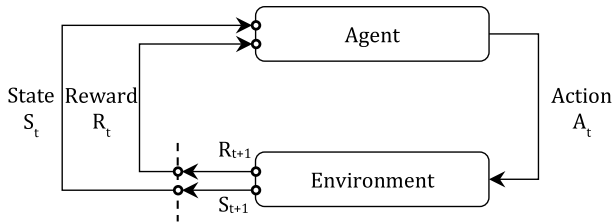
Although the AI techniques, such as meta-heuristic algorithms, neural networks, fuzzy methods, etc., have been recorded great performance, recently they are faced with some major challenges in the case of traffic control. They need a lot of prepared training datasets. In complex problems, their required computation highly increases. The rigid learned model of these methods cannot overcome the dynamic essence of the traffic control problem. The traffic data will change dynamically, and an ideal controller should adjust himself in these situations. Our proposed method is based on the deep reinforcement learning method. Reinforcement learning agent learns the appropriate policies for dynamic environments in the absence of the training dataset. The method also employs a priority controller for emergency vehicles such as ambulances, firefighters, and police cars to decrease their waiting time before traffic lights. Our agent's crossroad state space and its action space model are designed with an emphasis on emergency vehicles, which is considered less in this literature.

### 3 Reinforcement learning

Reinforcement learning is a branch of machine learning in which one or more agents learn in an environment to solve a problem using the experiences generated and the interactions between the agent and their environment [17]. One of the goals of artificial intelligence is to develop machines that behave like human beings. For this purpose, it is necessary to learn from the interactions with the environment, to make decisions, and to act properly. Part of artificial intelligence that is capable of autonomous and experiential learning is reinforcement learning [12]. Reinforcement learning learns behavior by interacting with an environment and receiving rewards [18]. Reinforcement learning has achieved some success in complex areas such as games, robotics, and traffic signal control [12].

The reinforcement learning cycle is shown in Fig. 1. The agent performs an action that results in a new situation in the environment and, accordingly, a reward is obtained, the reward and the new state are perceived by the agent and the cycle





**Fig. 1** Cycle of reinforcement learning

continues. During this cycle, he learns the optimal policy factor and becomes aware of the outcome of actions in the environment.

### 3.1 Reinforcement learning in traffic control

Traffic control is one of the most useful application areas for reinforcement learning techniques. In this framework, one or more independent agents operate to maximize the efficiency of traffic flow through one or more intersections controlled by traffic lights [12]. The use of reinforcement learning to control traffic lights for a variety of reasons is of interest [19]:

- A. In the case of correct learning, reinforcement learning factors can be adapted to different situations (e.g., road accidents, adverse weather conditions).
- B. Enhanced learning factors can be self-taught without supervision or prior knowledge of the environment.
- C. The agent only needs a simple model of the environment (which is essentially state of the art); however, the agent is trained using a system performance measure (i.e., reward).

In this study, reinforcement learning is used to solve the challenge, while the presence of emergency vehicles on the lanes is also important. The timing of the green phase is controlled by another flexible and dynamic method. The traffic light is the agent of this problem. Other key components of reinforcement learning are described below.

The state of the agent represents a state of the environment at time  $t$ , often represented by  $s_t$  [12]. The space state of the problem, if be small, may suffer from information deficits and, as a result, lack of information may impair learning, and if presented with too much detail, large and complex space, making it difficult to calculate and make it difficult to search for a solution and slow learning.

An agent's action is defined as interactive behavior with the environment. In the area of traffic control, the agent action is implemented with varying degrees of flexibility [17]. The dynamic scheduling of the agent's selective phase makes it more flexible and thus achieves optimal and dynamic traffic control.

The reward is usually used by the agent to understand the effect of the last action in the last state; it is usually defined as a function of some intersection efficiency

index, such as vehicle delays, queue lengths, waiting times, or overall throughput [12].

There are different algorithmic frameworks for reinforcement learning methods from different perspectives. One of them which is model-free and value-based is the Q-learning method. Q-learning is exploration-intensive. This means that no matter what discovery policy is followed, it will converge to the optimal policy, assuming that each state-action pair is visited indefinitely and the learning rate  $\alpha$  is appropriately reduced [20]. Q-Learning directly estimates the value of an action in one state, the so-called Q-value of the pair  $s, a$ , is called  $Q(s, a)$ . In the traditional Q-learning method, the agent uses a lookup table of  $s, a$  and repeatedly estimates the value of  $Q$  [21]. The update formula of Q-value is shown in Eq. 1:

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

where  $Q(s_t, a_t)$  is the value of the action at the state  $s_t$ . This equation involves updating the current Q-value with a learning rate coefficient  $\alpha$ . In parentheses, the expression  $r_{t+1}$  is the reward for the action  $a_t$  at  $s_t$ . The  $t + 1$  index is used to emphasize the temporal relationship between the action taken  $a_t$  and the subsequent reward. The term  $Q(s_{t+1}, a_t)$  represents the near future Q-value where  $s_{t+1}$  is the next state in the environment after the action of  $a_t$  at  $s_t$  has taken place. The  $\max_A$  statement means to choose the most valuable of the possible actions  $a_t$  at  $s_{t+1}$ . The coefficient  $\gamma$  is a discount factor that assumes a value between 0 and 1 and diminishes the importance of future rewards compared to immediate rewards [12]. The pseudo-code of Q-learning is presented in Algorithm 1.

---

#### Algorithm1 - Q-learning

---

Initialize  $Q(s, a)$  arbitrary

**Repeat** (for each episode)

Initialize  $s$ ,

**Repeat** (for each step of the episode)

Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)

Take action  $a$ , observation,  $s'$

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

$s \leftarrow s'$

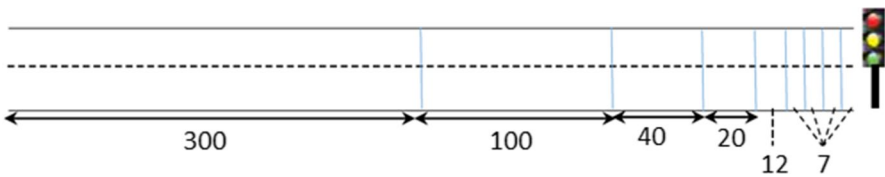
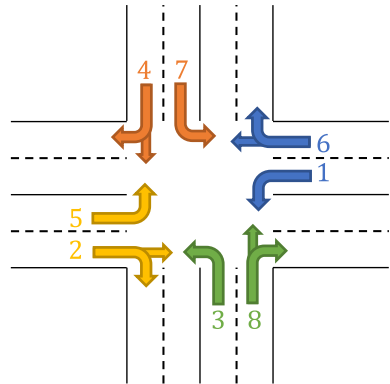
**Until**  $s$  is terminal

---

After initializing the Q-values with the desired initial value, the initial state for each episode is initialized. Then, for each step in the episode, an action is selected in the current state according to the policy, after which the action is taken, a new observation and reward are received, and the Q-value is updated according to its values and transferred to the new state.

While Q-table learning works well in small domains, many real-world problems have very large or continuous state space and action space, and therefore do not allow to count the  $s, a$  pair. One solution to this problem is the approximation function, in which supervised machine learning algorithms are used to approximate Q-function. In this case, the Q-value is no longer an input to the table, but a function whose parameters are determined by the learned weights. These weights are updated

**Fig. 2** Intersection and traffic flows



**Fig. 3** Discretization of one arm of intersection

by gradient descent methods, minimizing the mean squared error between the current estimate of  $Q(s, a)$  and the target, which is defined as the objective of the true Q-value of the pair  $s, a$  under  $\pi$  policy [21].

The  $\epsilon$  - greedy probability allows Q-learning to have the chance to randomly select an action in the action space that does not have the highest Q-value. Introducing greedy probability ensures the agent has a chance to explore the new environment (exploration). Without  $\epsilon$ -greedy probability, the Q-learning process will not tend to explore the new environment, eventually losing out on the opportunity to gain new experience with greater rewards. However, if the  $\epsilon$ -greed probability is too high, the stability of Q-learning decreases, as the algorithm discovery will hinder the solution in the known environment by experience [22]. Algorithm 2 is shown as the  $\epsilon$  - greedy algorithm.

---

**Algorithm 2 -  $\epsilon$  - greedy**

---

With probability  $1 - \epsilon$   
     Choose  $agrmax_a Q(s, a)$   
 With probability  $\epsilon$   
     Select random action

---

## 4 Implementation

For implementation, it is first necessary to define the key components for traffic control with reinforcement learning and then explain the settings of each. In this study, the environment is a typical intersection as shown in Fig. 2. To separate the flows, each one is assigned a number. The distance from the source to the traffic light is 500 m. The total simulation space is  $1000 \times 1000$  square meters. The traffic light is located at the center of the intersection to control traffic flows.

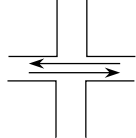
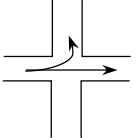
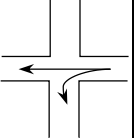
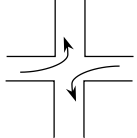
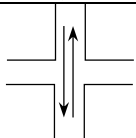
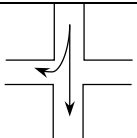
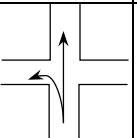
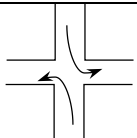
### 4.1 State space

In this problem, each arm is discretized from intersection inputs into cells that can determine the presence or absence of a vehicle within them. Figure 3 illustrates the state of an intersection arm. Between the beginning of the road and the intersection, there are 18 cells, 9 for one line and 9 for the other. So, there are 72 cells in all 8 intersections. Cells are not the same size. Farther from the intersection are longer cells. Traffic congestion and vehicle stoppages occur mostly at distances close to the intersection.

Choosing the length of each cell is not trivial; if the cells are too long, some cars may not be marked near the lane. If the cells are too short, the number of modes needed to cover the line increases, leading to higher computational complexity. The sum of the disaggregation figures is 500 m.

In addition to each of the 9 cells from the segmentation of the flow lines, another cell is considered. This cell is intended to identify the presence of emergency vehicles on each line. If there is an emergency car in each of the lanes, the cell will be equal to one, and else is zero. The proposed state space consists of 80 Boolean cells. This means that the number of possible modes is  $2^{80}$ .

**Table 2** Possible no-conflict pair flows

 flows 2,6	 flows 2,5	 flows 1,6	 flows 1,5
 flows 4,8	 flows 4,7	 flows 3,8	 flows 3,7

## 4.2 Action space

As shown in Fig. 2, each car at this intersection has three choices: straight, turn right, turn left. So, there are 12 different moves for the intersection. We consider that turning right is always open and cars can join right without any other traffic flows, so only 8 traffic flows remain in conflict, requiring timing and priority crossing.

At each intersection, several flows are in conflict and traffic lights are designed to ensure efficient and safe car scheduling and prioritization. Possible no-conflict pair flows in Table 2 show all pairs of possible flows that can cross the intersection at the same time.

During the simulation, the agent samples from the environment, and according to the policy, chooses an action, which is to select a phase between the options within possible no-conflict pair flows in Table 2.

## 4.3 Reward function

In this study, the reward function is the sum of the total waiting time, i.e., the accumulated total waiting time. If there is an emergency car in the environment, due to its higher importance, the total waiting time of all emergency vehicles will be multiplied by a numerical constant; thus, the waiting time of these vehicles will have a greater impact on the final reward rate than ordinary vehicles. The formula is used to calculate the total waiting time is shown in Eq. 2:

$$atwt_t = \sum_{veh=1}^n awt_{(veh,t)} + \beta \sum_{emr=1}^m awt_{(emr,t)}, \quad (2)$$

where  $awt_{(veh,t)}$  is the amount of time measured in second that the speed of a normal vehicle  $veh$  is less than 0.1 m/s from the time it enters the environment.  $n$  is the number of vehicles in the environment at time  $t$ .

The  $\beta$  constant is a factor considered to increase the latency effect of emergency vehicles.  $awt_{(veh,t)}$  is the amount of time measured in second that the speed of an emergency car  $emr$  is less than 0.1 m/s from the time it enters the environment.  $emr$  is the number of emergency vehicles in the environment at time  $t$ . So,  $atwt_t$  is the accumulated total waiting time at time  $t$  with a greater focus on waiting times for emergency vehicles. By this scale, the reward function is as follows in Eq. 3:

$$r_t = atwt_{t-1} - atwt_t, \quad (3)$$

where  $r_t$  represents the reward at time step  $t$ .  $atwt_t$  and  $atwt_{t-1}$  represent the accumulated total waiting times of all intersection vehicles in the time step  $t$  and  $t - 1$ , respectively.

It is noteworthy that the emergence of an emergency vehicle occurs in just a few moments of simulation. In the absence of an emergency vehicle or leaving the lane, Eq. 3 becomes Eq. 4:

$$\text{atwt}_t = \sum_{\text{veh}=1}^n \text{awt}_{(\text{veh}.t)}. \quad (4)$$

Thus, in addition to the public traffic on that line, special attention has also been paid to emergency vehicles, with delays affecting the total waiting time more than ordinary vehicles.

#### 4.4 Learning algorithm

The learning mechanism in this study is called deep Q-learning, which combines two commonly used methods in reinforcement learning: deep neural network and Q-learning as Eq. 5:

$$Q(s_t, a_t) = r_{t+1} + \gamma \cdot \max_A Q'(s_{t+1}, a_{t+1}), \quad (5)$$

that is, the reward  $r_{t+1}$  is the reward received after doing  $a_t$  at  $s_t$ . The expression  $Q'(s_{t+1}, a_{t+1})$  is the value of  $Q'$  for action  $a_{t+1}$  in the state of  $s_{t+1}$ , that is, after the state of action at  $s_t$ . As observed in Eq. 5, the coefficient  $\gamma$  is a small discount for future rewards compared to immediate rewards.

#### 4.5 Deep reinforcement learning

In a reinforcement learning program, the state space is usually very large, and it is impractical to discover and store each pair of states actions. Therefore, the Q-learning function is approximated using a neural network. The input of the neural network is a state; the state is a vector with 80 cells at any given moment. Network output is also a Q-value set of possible actions; there are 8 actions at any one time.

In this study, a fully connected neural network was used, which consisted of an 80-neuron input layer, 2 hidden layers with 128 and 64 neurons, each with a modified linear unit activator function (RELU), and an 8-neuronal output layer with a linear activator function. Each of the output neurons represents the value of one of the given states. A visual representation of the defined deep neural network is shown in deep neural network representation in Fig. 4.

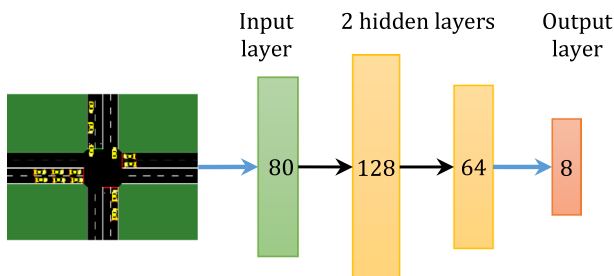


Fig. 4 Deep neural network representation

Experience replay [23] is a technique adopted during the training phase to improve agent performance and learning efficiency. This includes sending the information needed for learning in a random group of samples called batch to the agent, instead of sending the information that the agent collected during the simulation (often called online learning). The batch is usually derived from a data structure called memory that stores each sample collected during the training phase. An example of an  $m$  in the form of a quadrilateral is defined as Eq. 6:

$$m = \langle s_t, a_t, r_{t+1}, s_{t+1} \rangle, \tag{6}$$

---

**Algorithm 3 - Agent Training Pseudocode**

---

```

FOR ( $h: 1 \rightarrow H$ ) //  $H$  is the number of total episodes
    Memory  $\leftarrow m = \langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$ 
     $\mathcal{M} \leftarrow$  Random Batch (Memory)
     $\epsilon_h = 1 - \frac{h}{H}$ 
    FOREACH ( $m_i \in \mathcal{M}$ )
         $Q = \text{NeuralNet}(s_t, a_t)$  // Predict Q-value by neural network
        IF ( $\text{random} < \epsilon_h$ )
             $a_{t+1} = \text{Choose random action}$ 
        ELSE
             $a_{t+1} = \max_A Q(s_{t+1}, a_t)$ 
         $Q' = \text{NeuralNet}(s_{t+1}, a_{t+1})$  // Predict Q'-value by neural network
        Compute  $r_{t+1}$  with Equation 3
         $Q_{\text{new}} = r_{t+1} + \gamma \cdot Q'$ 
        Train NeuralNet with {input:  $(s_{t+1}, a_{t+1})$ , output:  $Q_{\text{new}}$ }
    
```

---

where  $r_{t+1}$  is the reward received after performing the action  $a_t$  at state  $s_t$ , which takes the environment to the next state  $s_{t+1}$ .

As mentioned, the replay information technique requires a memory, which is marked with a memory size and a batch size. The size of the memory shows how many samples can be stored in the memory and set to 50,000. The batch size is defined as the number of samples to be retrieved from the memory in a training instance and is set to 100. Each time a sample is produced during the agent activity, the training process takes place. The agent training pseudocode is shown in Algorithm 3.

The steps of the experience and training process can be mentioned as follows:

- (1) An  $m$ -sample containing the items listed in Eq. 6 is added to the memory.
- (2) A fixed number of samples (a batch) are randomly taken from memory.
- (3) For each instance in the batch, the following operations are performed:
  - a. Prediction of Q-values of  $Q(s_t, a_t)$  with the neural network, which is the values of actions in the  $s_t$  state.
  - b. Prediction of Q'-values of  $Q'(s_{t+1}, a_{t+1})$  with the neural network which is the values of actions in  $s_{t+1}$  mode.
  - c. An update of  $Q(s_t, a_t)$  that indicates the value of the specific action selected by the agent during the simulation. This value is rewritten using the Q-learning function

described in Eq. 5.  $r_{t+1}$  rewards for action at,  $\max_A Q'(s_{t+1}, a_{t+1})$ . The choice of the most expected reward for the future indicates that  $Q'(s_{t+1})$  is obtained using the prediction. That is the maximum value of the expected action of the agent, starting from  $s_{t+1}$ . This value is reduced by the degradation factor  $\gamma$ , which gives more importance to immediate reward.

- d. Neural network training. The input state is  $s_t$ . The desired output is the updated Q-values of  $Q(s_t, a_t)$ , which now includes the maximum expected reward for the future, thanks to the Q-value update.

When the deep neural network estimates the learning-Q function sufficiently, selecting the most valuable action in the current state results in the highest traffic efficiency.

#### 4.6 The exploration/exploitation tradeoff

A key issue in any reinforcement learning problem is the policy of choosing action during learning [12]. The agent acts exploratory to learn more, or exploitable to make a profit and to choose the most valuable action ever found. In the early stages of education, the agent does not know which action is more valuable. Therefore, at the beginning of the training, the agent must discover the results of the actions and not be afraid of losing efficiency. When an agent acquires a reliable knowledge of the results of the actions of a large part of the space of states, he must increase the frequency of for-profit actions.

This study uses the  $\epsilon$  - greedy exploration policy, which considers  $\epsilon$  probability for selective action and  $1 - \epsilon$  probability for for-profit selection in episode  $h$ . In Eq. 7, the value of  $\epsilon$  is defined:

$$\epsilon_h = 1 - \frac{h}{H}, \quad (7)$$

where  $h$  is the number of current learning episodes and  $H$  is the total number of episodes. The initial value of  $\epsilon = 1$  means that the agent acts exclusively exploratory. As education progresses, the factor increasingly exploits what it has learned, rather than benefiting it exclusively. The number of episodes starts from zero.

#### 4.7 Phase scheduling

One of the characteristics of the action of an agent is its flexibility, which determines the flexible scheduling of the selected phase. In this study, to determine the time of each selected phase by the agent, a dynamic method, which is also introduced in [24], has been used. This method assigns a variable between zero and constant Maxgreen to each phase and zero time for empty flows and maximum value for very dense flows. The Maxgreen value is considered a constant parameter to ensure a fair share of time between all traffic flows while competing. It is assumed that all traffic flows within the intersection are the main arteries.



The farthest distance that cars can travel in Maxgreen time is defined by the  $D_{\text{green}}$  variable. In other words, it can be considered a passable area during Maxgreen time. The length of this distance is calculated by Eq. 8:

$$D_{\text{green}} = \text{Maxgreen} \times T_{\text{speed}}, \quad (8)$$

where  $T_{\text{speed}}$  is the average speed of traffic flow. In situations where traffic is very congested, vehicles are crossing the intersection for the entire Maxgreen period, and in situations where the line traffic is almost empty, vehicles need less time than the Maxgreen time to cross the intersection. The best time for the traffic light to be green is calculated in the Bestgreen variable to reduce the wasted time. Bestgreen time for each phase is selected according to the traffic distribution in the corresponding line. This time has a threshold between zero and Maxgreen according to Eq. 9:

$$0 \leq \text{Bestgreen} \leq \text{Maxgreen}. \quad (9)$$

In other words, Bestgreen time is defined as the time required for the last vehicle to pass within the  $D_{\text{green}}$  range, which can be calculated using the location of the last vehicle during traffic and traffic speed in line. Equation 10 shows how to calculate Bestgreen for traffic lines:

$$\text{Bestgreen} = \frac{LV_{\text{Distance}}}{T_{\text{speed}}}, \quad (10)$$

where  $LV_{\text{Distance}}$  shows the distance of the last vehicle from the intersection in the  $D_{\text{green}}$  range (the distance that can be traveled in Maxgreen time) and  $T_{\text{speed}}$  is the average speed of the traffic flow. By dividing these two values, the time interval required for the last vehicle to pass within the range is obtained.

The Maxgreen value is set as a constant parameter in the initial implementation steps. After selecting each phase by the deep-Q learning algorithm, the length of that phase will be equal to the Bestgreen value.

**Table 3** Simulation parameters

Parameter	Value
Simulator	Python
Number of vehicles	200–1000
Number of emergency vehicles	1–4
Episode time	2000s (About 33 min)
Number of training episodes	100
Ma	60 s
Q-Learning Parameters	$\beta = 8, \gamma = 1$ (Default value in python library)
The map	4-leg traffic intersection

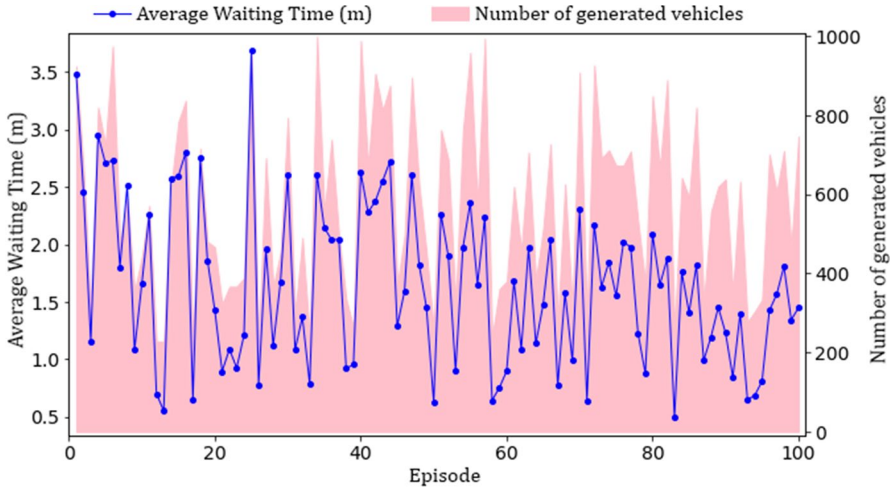


Fig. 5 Average waiting time (m) in 100 episodes

### 5 Simulation and results

SUMO (Simulation of Urban MOBility) [25] has been used to simulate traffic. SUMO is used to generate motion scenarios; Python is used to implement reinforcement learning and scheduling. Between 200 and 1000 vehicles are randomly produced at each stage of the run. Between one and four emergency vehicles are produced at a random time and path. It is supposed that there is a vehicular communication technology, which reports the emergency vehicles type, location, speed, and target destination to the traffic lights agent. The agent is trained in several episodes

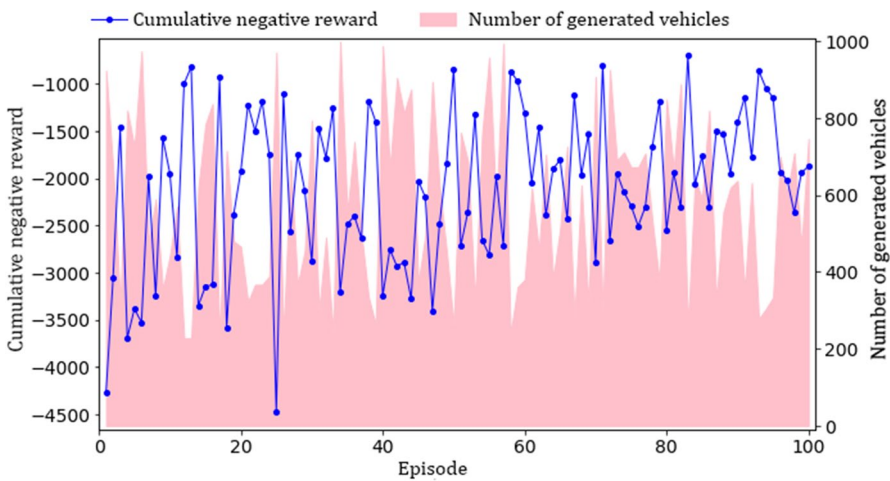


Fig. 6 Total negative reward in 100 episodes

and gains experience. Each episode involves completing a scenario wholly. The duration of each episode is 2000s, approximately 33 min. 100 episodes are used for training, which is the result of 200,000 s of simulation and is equivalent to more than 55 h or more than two full days of training. In other words, in each episode, which consists of 2,000 s, at least once every 10 s, the input vector is formed and the process of training and changing the traffic light takes place, which is at least 200 vectors in one episode, equivalent to at least 20,000 vectors. The efficiency scales used are described below:

- *Total negative reward* The sum of all the negative rewards gained during all the actions taken during the simulation in a complete episode. Each time the reward is negative, the value is added and this is done in a complete episode.
- *Average Delay* The average delay of vehicles per second during the completion of an episode.

In all tests, the Maxgreen parameter is 60 s and the  $\beta$  parameter is 8 and  $\gamma$  is 0.75. More than one emergency vehicle has been produced for less than 600 vehicles. The parameters that we set in the tested experiments are presented in Table 3.

The results of 100 episodes of learning the agent are extracted according to performance scales. Figure 5 shows the average delay of vehicles in 100 episodes. The horizontal axis shows progress in the episodes and the vertical axis on the left, the average delay in each episode, the vertical axis on the right shows the number of vehicles produced in each episode.

The average delay in progress in the number of episodes is decreasing; in other words, it is declining, because the more episodes that take place, the more and more the agent learns and experiences, and the more intelligent and measured the agent will act. In each episode, the number of cars is randomly generated and displayed

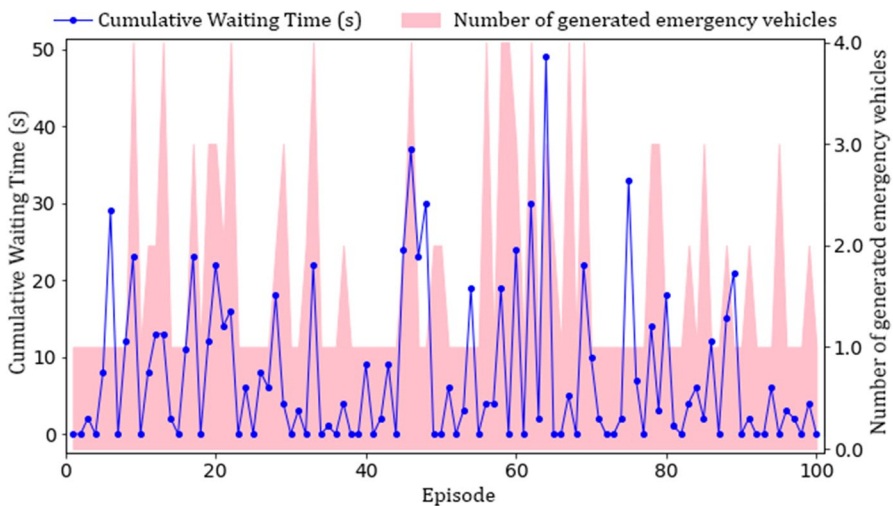


Fig. 7 Total negative reward of emergency vehicles

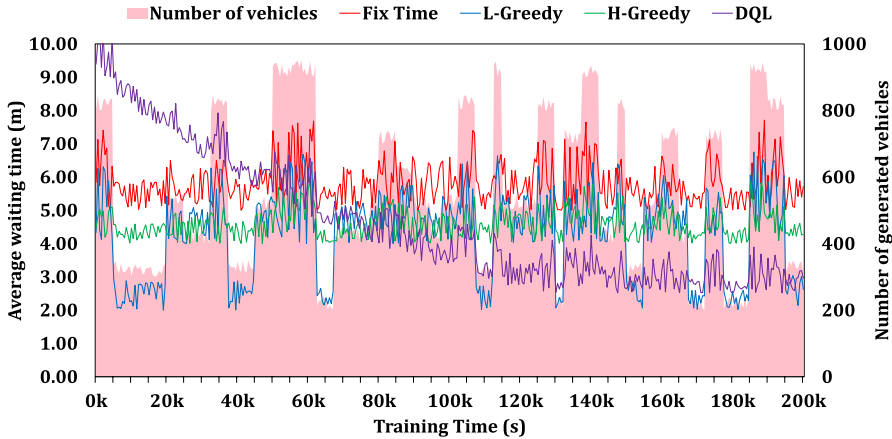


Fig. 8 Average waiting time (m) for fixed-time, L-greedy, and H-greedy compared to the DQL

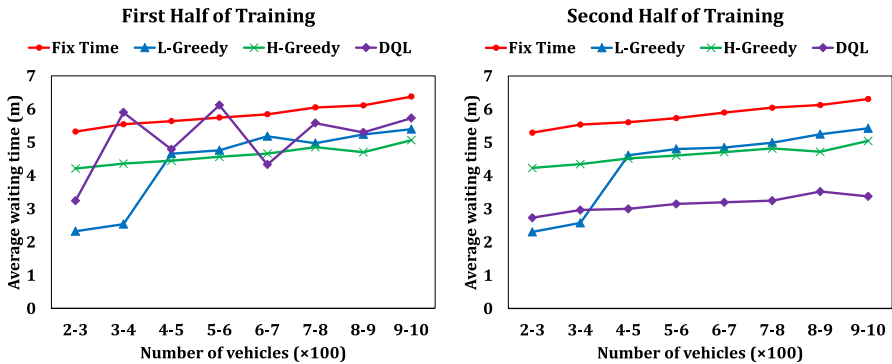


Fig. 9 Mean of the average waiting time (m) for fixed-time, L-greedy, and H-greedy compared to the DQL categorized by the number of vehicles

with the help of the vertical axis on the right. In each episode, both the average delay curve and the number of cars show almost the same ups and downs, meaning that in an episode with a large number of cars, the average delay is increased and if it is less, the average delay is also less. The high number of cars causes congestion and delays. However, with the increase in episodes, even in the high number of cars, the average delay is significantly different from the same number of cars in the introductory episodes.

Figure 6 shows the total number of negative rewards during the 100 episodes shown in Fig. 5. In each episode, each time a reward is received during the agent’s action, it is checked that if this value is negative, it is added up and stored to measure performance. The horizontal axis is the agent’s progress in the episodes, the vertical axis on the left is the total negative rewards of the agent in each episode, the vertical axis on the right is the number of vehicles produced in each episode.

The total negative rewards are closer to zero as they progress through the episodes. This upward trend shows that the agent has gained fewer negative rewards as the episodes increase, so learning the agent is increasing and the agent errors are reduced. Both the total negative rewards curve and the number of cars are almost inversely proportional to each other; in other words, at increasing points, the number of cars, the negative reward points are farther than zero; it is clear that in high congestion, more complex traffic management and the probability of error and negative results are higher. In the final episodes, this value is reduced even at high traffic densities, indicating proper traffic control.

Figure 7 shows the number and total delay of emergency vehicles in 100 episodes in average waiting time (m) in 100 episodes shown in Fig. 5. The total delay is the sum of the delays of emergency vehicles throughout the episode. The horizontal axis is the agent progress in the episodes, the vertical axis on the left is the total delay of the emergency vehicles in each episode, and the vertical axis on the right shows the number of emergency vehicles produced in each episode. In the early episodes, the total delay was high, but with the progression of the episodes and reaching the final episodes, the total delay was reduced. Usually, in places where the number of emergency vehicles is high, the total delay is more than the points with a fewer number of emergency vehicles. A significant decrease from episode 60 onwards and more zero values are seen.

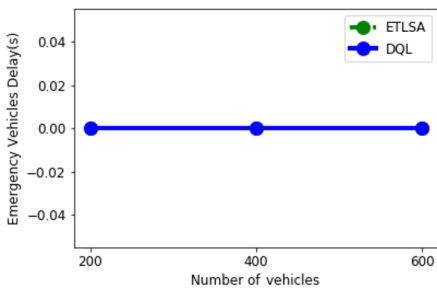
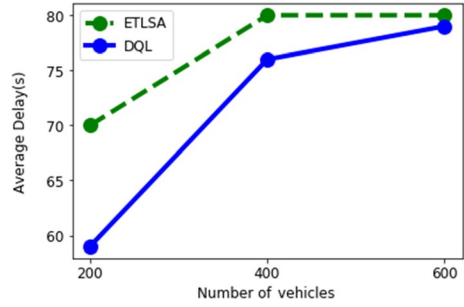
## 5.1 Comparison with state of the art

To analyze the performance of the proposed model, we compare our deep Q-learning (DQL) model with three state-of-the-art solutions. In the first solution, a fixed period has been set for traffic lights. Two greedy techniques also have been applied for comparison. In the L-greedy technique, the agent checks the number of the vehicles which are waiting before traffic lights and enqueues them in a priority queue in order to least number. The green light will assign to the head of the priority queue. The H-greedy is the same as the L-greedy, except that the order is in reverse, and the highest number will be placed in the head of the queue. The priority of the green light assignment is higher for the heavy traffic line. The comparison result is shown in Fig. 8. Again, the left vertical axis is the average waiting time for the vehicles, and the right vertical axis is the number of generated vehicles, randomly.

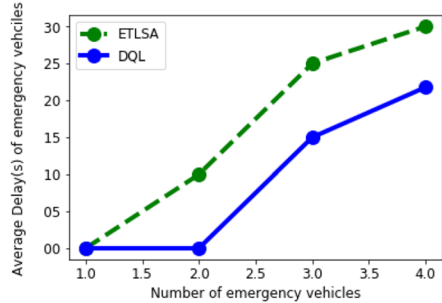
As expected, at the beginning of the training process, the agent does not perform well because of its random selections and records a high waiting time, even higher than the simple fix period technique. But after some steps, the agent learns the policies; in the middle of the training time, it achieves a good and consistent performance. After 100,000 s learning, the DQL average waiting time is lower than other techniques, except for the L-greedy in low traffic flow (less than 300 vehicles).

For more clarity, we classified the number of vehicles into 8 classes. The mean of the average waiting time for each technique according to each class is shown in Fig. 9. The result is divided into 2 timespans: the first half of the training time from 0 to 100,000 s, when the agent tries to learn the policies, and the second half of the training time, when the agent has been learned.

**Fig. 10** Total average delay of vehicles



**(a)** Delay of the emergency vehicle



**(b)** Average delay of emergency vehicles

**Fig. 11** **a** Delay of the emergency vehicle. **b** Average delay of emergency vehicles

The ups and downs of the DQL graph in the first half of the training indicate the random behavior of the agent and its attempts to learn suitable policies. In the second half of the training, the proper learning of the agent could be proved by its less waiting time. (Note that the horizontal axes in Fig. 9 do not show the time, but they are the frequency of the vehicles.) As mentioned before, just the L-greedy technique has better performance than our DQL in a light traffic flow in which the number of vehicles is less than 300. Assigning the higher priority to the light traffic line to cross the intersection decreases the waiting time significantly. But in the case of heavy traffic, the more and faster changing the traffic light phases will increase the waiting time. In the H-greedy technique, assigning a high priority to heavy traffic causes starvation in fewer traffic lanes. As a result, the green light does not assign to the lines with fewer vehicles and their waiting time significantly increases. In summary, our DQL model has 37% less waiting time than the fixed-time method, 34% than L-greedy in heavy traffic flow, and 33% than H-greedy. The L-greedy has 11% better performance than the proposed DQL model in light traffic flow.

## 5.2 Comparison to similar works

To evaluate the efficiency, the proposed agent has been compared with the research [24]. The algorithm [24] is called ETLISA, and the proposed agent of this research is called DQL. In ETLISA, the phases of traffic lights are selected according to a deterministic and computational algorithm, according to the density and speed of vehicles to cross the intersection. Also, if the line contains an emergency vehicle, it will have a higher priority and will be green sooner to cross the intersection. In ETLISA, the proposed algorithm requires a lot of exact information about traffic, such as the total density of lines, the exact distance of the emergency vehicle from the intersection and its exact speed, the exact density of the front area of the emergency vehicle. In addition to the above, in ETLISA, the scheduling and determination of phases are done at the end of a cycle (periodic sequence of phases), i.e., a cycle is planned and given to the traffic light, and by the end of that cycle, the calculation and change in the schedule and the phasing do not take place. But the traffic density of the lines changes momentarily, and by the end of a phase that is part of a cycle, it may change rapidly, and considering these changes can be effective in scheduling and phasing the lines. But in the current study, the proposed agent, after selecting each phase, examines the new conditions and rewards and then selects a new phase. No precise information is required except for the presence or absence of vehicles in the cells designated for the lines and the presence or absence of an emergency vehicle. Two data are necessary for scheduling, the average speed of the cars and the last car within the selected area. It is supposed that vehicular communication technology will send this information to the agent. In general, the proposed agent has acquired its intelligence in the form of self-learning and self-regulation and does not require manual intervention and computational definitions.

A few final episodes after 100 episodes have been considered for testing the final model, in which the number of cars and emergency vehicles is determined and the process of experience replay is disabled. 200, 400, and 600 ordinary vehicles and one emergency vehicle were randomly produced, and the total average delay of vehicles was measured.

As shown in Fig. 10, the total average delay with the Q-learning agent is reduced by about 2–16%. The trend of both methods is upward, because it is obvious that with the increase in the number of vehicles, the density increases and more congestion causes an increase in the average delay of vehicles.

The delay of the simulated emergency vehicle in all the number of vehicles of the above scenarios is zero, and this number is the same in both methods as shown in Fig. 11a.

Fig. 11b shows the average delay of 600 randomly generated vehicles and one to four emergency vehicles with random routes are produced and simulated. More than four emergency vehicles are rare in the real world. The trend of both methods is upward, and with the increase in the number of emergency vehicles and making the emergence of exceptional conditions, the average delay for emergency vehicles is increasing. As can be seen in the chart, the average delay of emergency vehicles with Q-Learning agents up to 2 emergency vehicles is zero, and then, it is spaced from zero and higher values are included in the chart. This value is 27% to 40% lower than the ETLISA method for DQL.

## 6 Conclusion

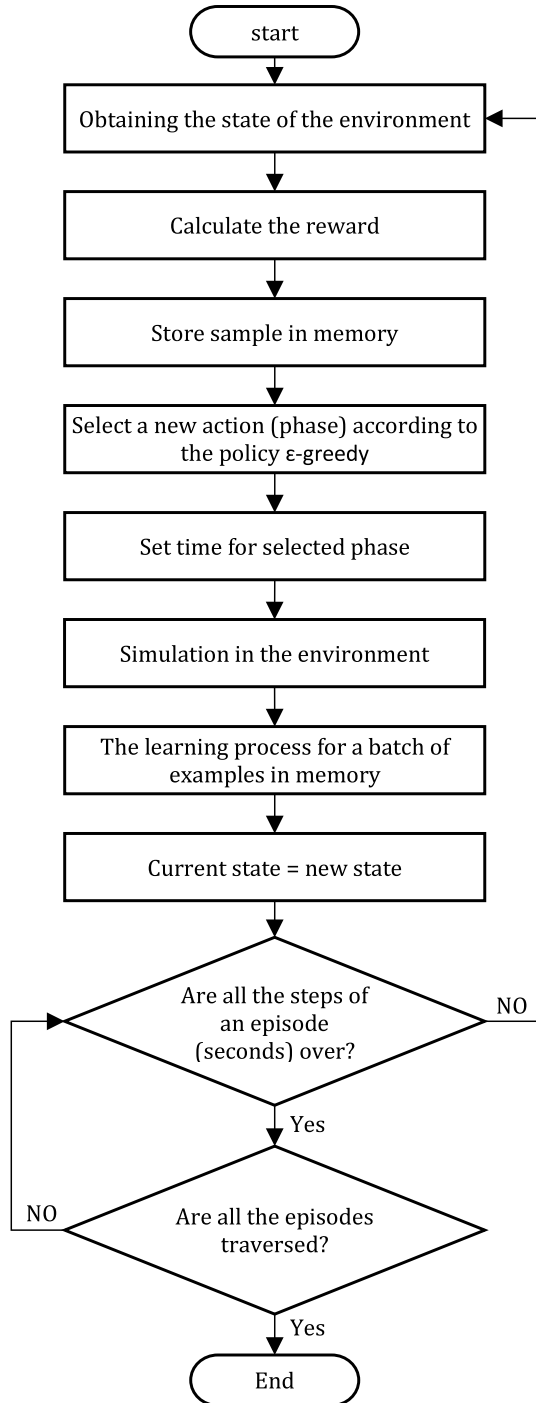
In this study, deep reinforcement learning, especially the Q-learning method, has been used to control traffic at an intersection. The traffic light operates as a controlling agent at the intersection. An intelligent trained agent with the Q-learning method and a deep neural network learns the control policies to manage the traffic at an intersection. The formal model of crossroad traffic with its state space, action space, and reward function is designed with an emphasis on emergency vehicles. The agent dynamically schedules the traffic lights phase, to achieve less waiting time and emergency vehicles delay. The results show an effective reduction in delay, compared to the other state-of-the-art methods. The total average of delays of one emergency vehicle among 200–600 cars decreased about 2–16%, while with more emergency vehicles, there is a 27–40% decrease in average delay for emergency vehicles.

As future work, it is possible to increase routes and implement a network of roads instead of just one intersection or more intersection types such as T intersections. The combination of the intelligent phasing of traffic lights with a proper emergency vehicle routing algorithm is an interesting issue and will increase the speed of reaching the destination and will reduce service delays. We tested the system in a simulated traffic model, not in the real world. It is helpful to simulate it for some city and highway maps, as well as to import other real-world items such as pedestrians, bicycles, and motorcycles. It is interesting to examine how these items affect the system's policies and how the agent adjusts his policies to overcome these side-effects. The last suggestion is to examine other methods and algorithms of reinforcement learning, or considering other reward functions, for future research.

## Appendix

See Fig. 12



**Fig. 12** Proposed reinforcement learning flowchart

## Declarations

**Conflict of interest** The authors declare that there is no conflict of interest.

## References

1. Litman T (2013) Congestion Costing Critique: Critical Evaluation of the “Urban Mobility Report”
2. Nellore K, Hancke GP (2016) A survey on urban traffic management system using wireless sensor networks. *Sensors* 16(2):157
3. Grégoire P-L et al (2007) Urban traffic control based on learning agents. in 2007 IEEE Intelligent Transportation Systems Conference.. IEEE
4. Liu Z (2007) A survey of intelligence methods in urban traffic signal control. *IJCSNS Int J Computer Sci Netw Secur* 7(7):105–112
5. Alam J, Pandey M and Ahmed H (2013) Intelligent Traffic Light Control System for Isolated Intersection Using Fuzzy Logic. in Proceedings of the Conference on Advances in Communication and Control Systems-2013. Atlantis Press
6. Homaei H, Hejazi S, Dehghan SAM (2015) A new traffic light controller using fuzzy logic for a full single junction involving emergency vehicle preemption. *J Uncertain Syst* 9(1):49–61
7. Ben Arbi I (2019) A Survey on Intelligent Urban Road Traffic Control Systems. (Research Proposal)
8. Singh L, Tripathi S, Arora H (2009) Time optimization for traffic signal control using genetic algorithm. *Int J Recent Trends Eng* 2(2):4
9. Park B, Messer CJ, Urbanik T (1999) Traffic signal optimization program for oversaturated conditions: genetic algorithm approach. *Transp Res Rec* 1683(1):133–142
10. Di Febraro A, Sacco N (2005) Freeway traffic control based on neural network estimation. *IFAC Proceedings Volumes* 38(1):66–71
11. Dhingra G et al (2019) Traffic management using convolution neural network. *Int J Eng Adv Technol* 8(5S):8471
12. Vidali A, Crociani L, Vizzari G, Bandini S (2019) A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management. In WOA Jun (pp. 42–50).
13. Mousavi SS, Schukat M, Howley E (2017) Traffic light control using deep policy-gradient and value-function-based reinforcement learning. *IET Intel Transport Syst* 11(7):417–423
14. Gao J et al (2017) Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network. arXiv preprint <http://arxiv.org/abs/1705.02755>
15. Genders W and Razavi S (2016) Using a deep reinforcement learning agent for traffic signal control. arXiv preprint <http://arxiv.org/abs/1611.01142>
16. Li L, Lv Y, Wang F-Y (2016) Traffic signal timing via deep reinforcement learning. *IEEE/CAA J Automatica Sinica* 3(3):247–254
17. Vidali A, Crociani L, Vizzari G, Bandini S (2019) A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management. In WOA Jun (pp. 42–50).
18. Lison P (2015) An introduction to machine learning. Language Technology Group: Edinburgh, UK,
19. Yau K-LA et al (2017) A survey on reinforcement learning models and algorithms for traffic signal control. *ACM Comput Surv (CSUR)* 50(3):34
20. Wiering M, Van Otterlo M (2012) Reinforcement learning. *Adapt Learn Optim* 12:3
21. Van der Pol E (2016) Deep reinforcement learning for coordination in traffic light control. Master’s thesis, University of Amsterdam
22. Chin YK et al (2011) Q-learning based traffic optimization in management of signal timing plan. *Int J Simul Syst Sci Technol* 12(3):29–35
23. Lin L-J (1993) Reinforcement learning for robots using neural networks., Carnegie-Mellon Univ Pittsburgh PA School of Computer Science
24. Younes MB, Boukerche A (2018) An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems. *Wireless Netw* 24(7):2451–2463
25. Behrisch M et al (2011) SUMO—simulation of urban mobility: an overview. in Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation. ThinkMind.

26. Chu H, Liao Y, Chang L, Lee Y (2019) Traffic Light Cycle Configuration of Single Intersection Based on Modified Q-Learning. *Appl Sci* 9:4558–4577
27. Kumar N and Rahman (2019) Deep Reinforcement Learning with Vehicle Heterogeneity Based Traffic Light Control for Intelligent Transportation System, *IEEE International Conference on Industrial Internet*, pp. 28–33
28. Wei H, Zheng G, Yao H and Li Z (2018) IntelliLight: a Reinforcement Learning Approach for Intelligent Traffic Light Control, *ACM International Conference on Knowledge Discovery & Data Mining*, pp. 2496–2505
29. Bouktif S, Cheniki A, Ouni A (2021) Traffic signal control using hybrid action space deep reinforcement learning. *Sensors* 21:2302–2317
30. Maske H, Chu T, Kalabić U (2020) Control of traffic light timing using decentralized deep reinforcement learning. *IFAC-PapersOnLine* 53(2):14936–14941

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.