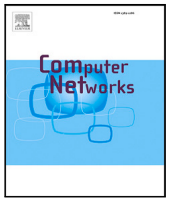
Computer Networks 234 (2023) 109934

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Review article

A survey on how network simulators serve reinforcement learning in wireless networks

Serap Ergun [a,b,*], Ibrahim Sammour [b], Gerard Chalhoub [b]

[a] Department of Computer Engineering, Isparta University of Applied Sciences, Isparta, 32260, Turkey
[b] LIMOS, University of Clermont-Auvergne, Clermont-Ferrand, 63170, Aubière, France

ARTICLE INFO

Keywords:
Wireless networks
Reinforcement learning
Network simulators

ABSTRACT

Rapid adoption of mobile devices, coupled with the increase in prominence of mobile applications and services, resulted in unprecedented infrastructure requirements for mobile and wireless networks. To improve user experience, future 5G and wireless network systems evolve to support increased mobile traffic, real-time precision analysis, and adaptable network resource management. As mobile environments become more complex, heterogeneous, and evolving, these tasks become more difficult. In order to solve these problems, many researchers rely on reinforcement learning. The success of reinforcement learning stems from its support for new and powerful tools that solve problems. Nodes mobility, instability of wireless connections, the coexistence of multiple wireless technologies, and resource sharing among users are a few examples of what makes a wireless network a dynamic system. Learning, which is the main feature of reinforcement learning, enables wireless nodes to adapt to the dynamics of the system over time. For the learning to be efficient, it should be done over realistic and varied conditions. This is where network simulation tools can be useful. Network simulators are extensively used when it comes to studying wireless network protocols. They offer the advantage of scaling up scenarios at minimum cost and the ability to test many possible configurations quicker under a controlled environment. The main purpose of this survey is to show how network simulators help in developing reinforcement learning techniques in wireless networks. We emphasize how these tools can be used in the learning process and which problems they can solve. In the end, we discuss open issues related to this topic and highlight some best practice guidelines when it comes to mixing network simulators, reinforcement learning, and wireless protocols.

1. Introduction

Wireless networking is facing a huge transformation changing the manner in which IP (Internet Protocol) devices connect and share data via radio frequencies and light waves. These technologies also enable the deployment of IoT (Internet of Things) networks, which integrate a diverse set of wireless devices ranging from smartphones to unmanned aerial vehicles, detectors, smart cars, wearable devices to virtual reality devices [1,2]. With the massive deployment of IoT and the fifth-generation (5G) mobile networks, there has been a rapid increase in the demand of users and data traffic for broadband wireless access networks over the last few years [2,3]. This requires the different wireless technologies used to adapt to different types of applications and to offer performance that answers different needs, such as high throughput, low latency, low packet loss, low energy consumption, and so on. Heterogeneous Networks (HetNets) are emerging, in which multiple different technologies and solutions for different needs are combined. It is possible to deploy these technologies on the same devices in the same areas and for the same or different purposes. Controlling heterogeneous infrastructures, on the other hand, is a challenging issue, especially when applied to large IoT systems. It needs the use of a complicated system to ensure their functionality and optimize data flow [4,5]. By allowing User Equipment (UE) to use several Radio Access Technologies (RATs) simultaneously inside a unified access network, 5G communications systems ensure timely and appropriate RATs within a unified access network. 5G enables a wide range of connections, along with the Internet of Things (IoT), Machine-to-Machine (M2M), Device-to-Device (D2D), and Vehicle-to-Everything (V2X) [6]. As a result, mobile users are confronted with even more heterogeneous connectivity options, which complicates efficient decision-making when dynamically selecting a network [7,8].

Many upcoming wireless networks aim to employ protocols and techniques that call for Machine Learning (ML) concepts across the

* Corresponding author at: Department of Computer Engineering, Isparta University of Applied Sciences, Isparta, 32260, Turkey.
E-mail address: serapbakioglu@isparta.edu.tr (S. Ergun).

https://doi.org/10.1016/j.comnet.2023.109934
Received 2 February 2023; Received in revised form 18 June 2023; Accepted 10 July 2023
Available online 15 July 2023

wireless core and edge infrastructure in order to provide proper ultra-reliable low-latency high data rate communications and widespread connectivity for network devices [6,9,10]. ML is a field of study that enables devices to learn to solve new problems without being explicitly programmed to solve them. ML techniques are broadly categorized as supervised learning, unsupervised learning, and Reinforcement Learning (RL) [1,11]. The aim of supervised learning is for the learning agent to learn a basic policy for linking inputs to outputs from a labeled data set of sample inputs and their expected results. Unsupervised learning requires no labeled data, and the agent attempts to find structures from the given input [12]. In RL, the agent incessantly connects with its environment, attempting to produce a good decision-making policy based on the reward/cost feedback provided by the environment [1,13].

RL has emerged as one of the most important research aspects of ML in the past 20 years and proved to have a major influence on the advancement of artificial intelligence (AI). RL is a learning process wherein an agent can make decisions on a routine basis, examine the results, and then modify its strategy instantly in order to find the optimum policy [2,4]. RL is a type of goal-oriented learning that teaches models how to achieve specific goals while maximizing results over time, rewarding positive behavior while punishing undesirable behavior. RL agents can perceive and interpret their environment, as well as act and learn through trial and error [14]. The goal of the agent is to maximize its cumulative reward, also known as the expected return. Different RL methods produce different behaviors in order for the agent to achieve its goal.

ML has piqued the interest of wireless network researchers due to its demonstrated efficacy in addressing complex multi-domain problems with near-optimal results [15]. RL, in particular, is extremely coveted for wireless networks due to the following benefits according to [9]:

- RL is capable of rapidly solving a variety of optimization issues. It offers computation-efficient solutions to a wide range of optimization problems, including Markov Decision Process (MDP), game problems and non-convex optimization problems that are difficult to solve.
- RL does not rely on precise environment modeling. Through experimenting in an interactive environment, RL agents can learn indirect information about network dynamics from raw high-dimensional data. By doing so, RL can then learn correlations between different factors.
- Individual network units can develop their own control policies using local data and without packet forwarding using distributed multi-agent RL. Relying on local learning and decision-making helps decrease the complexity and the overhead of the solution and increase the reactivity of the system.

ML requires very large data sets to train on. Ideally, these data sets must be comprehensive and of high quality for the system to work properly. However, in the creation of new data, algorithms need sufficient time to fulfill their objectives accurately and to be able to converge toward them adequately. Of course, doing all of this needs huge resources. Being able to interpret the results produced by the algorithms accurately and descriptively is another challenge. Since not every algorithm will fit every system, choosing the appropriate algorithm for the system is extremely important [16]. During the learning process, even if certain convergence can be reached, the RL algorithm must learn and discover the whole real and complex system. In some cases, it can be difficult for a standard RL algorithm to offer an effective solution [17].

For the learning to be efficient, it should be done over realistic and varied conditions. This is where network simulation tools come in handy. Network simulators are extensively used when it comes to studying wireless network protocols. They offer the advantage of scaling up scenarios at minimum cost and the ability to test many possible configurations quicker under a controlled environment. Network simulators help to emulate the way a network operates. They do this by defining modules that represent network protocols, radio interfaces, wireless channels, and so on. They are one of the leading test and evaluation methods in the field of general computer networks and have an important place in academia and industry [18,19]. The simulation tools and results create an environment where those concerned in these fields can experience, test, compare and improve their proposals. In [18–20], some overviews of the most popular network simulators for modeling wireless network technologies can be found.

The introduction of tools such as NS-3 Gym has simplified the integration of machine learning techniques with network simulators. NS-3 Gym [21] provides a flexible and convenient interface between NS-3 simulator and reinforcement learning algorithms, allowing researchers to easily explore and optimize network protocols. Similarly, Tensorflow [22] has a C++ Application Programming Interface (API), which enhances the OMNeT++ simulator with machine learning capabilities by offering a framework that can incorporate various ML algorithms. These tools have been successfully utilized in numerous networking studies, such as in [23] to optimize Wi-Fi rate adaptation decisions, in [24] to enhance the performance of vehicular networks by minimizing handover latency and packet loss, and in [25] to enhance resource utilization and user satisfaction in heterogeneous networks.

A few review studies that have been published in the literature could be helpful. For instance, authors in [9] provide a thorough introduction to the application of Artificial Neural Networks (ANN) in wireless networks and illustrate their potential advantages over more conventional approaches. The various ANN structures and their applicability in diverse wireless network settings are explored in this paper. With an emphasis on machine learning in wireless sensor networks (WSNs), a different study with the designation [26] has a wider scope. In-depth descriptions of several machine learning techniques, methods, and applications in WSNs are given in this work. Along with the difficulties and unresolved research questions in this area, it addresses several strategies like clustering, classification, and regression. Both studies offer a thorough overview of various machine learning techniques and methods that can be used in these circumstances. Though both studies focus on machine learning in wireless networks, they take different approaches to the subject. On the other hand, our survey focuses primarily on how network simulators are used to train, validate and evaluate reinforcement learning techniques in wireless networks.

When it comes to RL, network simulation tools can be used to represent the environment that the agent interacts with in order to learn. Thus, the accuracy of the learning model will very much depend on the reliability of network simulators in offering realistic behavior to the agent. Simulation tools offer great flexibility in making it possible for the agent to interact with any type and use case of the environment. Whether it is during the offline training, online learning, validation, or evaluation phases, the main aim of this paper is to present a survey on how network simulators have been used while studying RL in wireless networks. We summarize recent work that has been done on the use of RL in wireless networks and how network simulators helped in the process. We only included works from the past 3 years in order to highlight the most recent proposals. The study also closes a research gap by summarizing common design challenges regarding how and to what extent these methods can be applied to network simulators.

The remainder of this paper is organized as follows. Section 3 summarizes recent proposals that used network simulators in the process of RL. It is subdivided into 3 main subsections depending on the simulation tool that is used. At the end of each subsection, we added a summary table that highlights the main features of each cited proposal. In Section 4, we discuss open issues when dealing with RL for wireless networks and what are the main challenges that still need attention. And finally, in Section 5, the conclusion and outlook part is given.

2. Reinforcement learning functional description

RL is a type of machine learning in which an agent learns to behave in an environment by performing actions and receiving feedback in the form of rewards or penalties. The key idea behind reinforcement learning is to enable an agent to learn from its own experiences rather than relying on a pre-defined set of rules. In other words, RL enables an agent to learn how to behave through trial and error. RL consists of the following components:

- Environment: the environment is the external world in which the agent operates. It can be anything from a simulation to a physical environment, such as a robot in an agricultural field. The environment provides the agent with *observations* and *rewards* based on its actions.
- Agent: the agent is the entity that interacts with the environment. The agent performs actions in the environment based on its *observations* and receives *rewards* or *penalties* based on the outcome of its actions.
- State: the *state* is the current environment situation that the agent observes. The *state* can be a set of indicators that describe the environment. A state space is the set of all possible set of indicators an environment can have.
- Action: the *action* is the decision made by the agent based on the current state of the environment. It is the output of the decision-making process of the agent. Each new action would have an impact on the next state of the environment. An action space is the set of all possible actions an agent is able to take.
- Reward: the *reward* is the feedback the agent receives from the environment based on its actions. It is a scalar value that indicates the quality of the action taken by the agent.

The goal of reinforcement learning is for the agent to learn a policy, a function that maps states to actions. The policy determines the actions of the agent in each state to maximize its cumulative reward over time. The expected cumulative reward, which is the total rewards the agent has gained over time, represents the quality of the current policy. The policy is updated based on the rewards received after each action.

2.1. Key concepts of RL

Key concepts in RL include the Bellman equation, state-value functions, and action-value functions:

- Bellman equation: it describes the relationship between the value of a state and its successor states. According to this equation, the value of a state is the immediate reward obtained in that state plus the discounted value of the best possible future rewards. It is given in Eq. (1):

$$V(s) = R(s) + \gamma \cdot \max_a \left\{ \sum_{s'} P\left(s' \mid s, a\right) \cdot V\left(s'\right) \right\} \tag{1}$$

  where $V(s)$ represents the value of state s, $R(s)$ is the immediate reward obtained in state s, γ is the discount factor that determines the importance of future rewards, $P(s'|s,a)$ is the probability of transitioning to state s' given action a in state s, and $\max_a$ denotes the maximum value over all possible actions.
- State-Value function: it estimates the expected return from a specific state under a given policy. It represents the long-term cumulative reward expectation of the agent by following that policy. It is given Eq. (2):

$$V_\pi(s) = \mathbb{E}\left[R(t+1) + \gamma \cdot V_\pi(s(t+1)) \mid s(t) = s\right] \tag{2}$$

  where π represents the policy, E denotes the expectation operator, $R(t+1)$ is the immediate reward obtained at time t + 1, and s(t + 1) represents the state at time t + 1.
- Action-Value function: it estimates the expected return from a state–action pair under a given policy. It represents the long-term cumulative reward expectation of the agent by taking a particular action in a specific state and following the policy. It is given in Eq. (3):

$$Q(s,a) = \mathbb{E}\left[R(t+1) + \gamma \cdot \max_{a'} Q\left(s(t+1), a'\right) \mid s(t) = s, a(t) = a\right] \tag{3}$$

  where a′ represents the possible actions in the next state s(t + 1), and $\max_{a'}$ denotes the maximum value over all possible actions in the next state.

2.2. Policy update techniques

RL algorithms use various techniques to update the policy, such as value-based, policy-based, and actor–critic methods.

2.2.1. Value-based methods

Value-based methods rely on the estimation of the state–action value function, which is a function that estimates the expected cumulative reward for taking a particular action in a particular state. This value is known as Q-Value. The agent updates its policy by selecting the action with the highest state–action value, which is known as the Q-Value. What follows is a list of the pros and cons of value-based methods:

- They can learn in continuous action spaces by using function approximation methods, such as neural networks, to represent the Q-function, such as in DQN (Deep Q-Networks).
- They are not suitable for handling large state and action spaces as they can lead to high memory requirements and computational costs.
- They can suffer from overestimating or underestimating Q-values, which can lead to sub-optimal policies.

Some examples of value-based methods are Q-Learning and Deep Q-Learning. Q-learning and Deep Q-learning focus on learning an optimal action-value function called the Q-function. Q-learning is a model-free algorithm that iteratively updates Q-values based on observed rewards and state transitions. The update equation for Q-learning is as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \left[r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)\right] \tag{4}$$

where $Q(s,a)$ represents the Q-value for taking action a in state s, α is the learning rate, r is the immediate reward, γ is the discount factor, s' is the next state, and $\max_{a'} Q(s',a')$ represents the maximum Q-value among the available actions in the next state.

Deep Q-learning extends Q-learning by utilizing deep neural networks to handle high-dimensional state spaces. It involves training a DQN to approximate the Q-values. The DQN architecture typically includes convolutional and fully connected layers. The loss function for DQN is based on the Mean Squared Error (MSE) between the predicted Q-values and the target Q-values:

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \cdot \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta)\right)^2\right] \tag{5}$$

where $L(\theta)$ represents the loss, θ denotes the parameters of the DQN, θ^- represents the target network parameters, and the expectation is taken over a batch of experiences sampled from the replay buffer.

Deep Q-learning uses a separate target network that has delayed updates to provide more stable and reliable targets for training. The DQN is trained iteratively by optimizing the loss function using techniques such as gradient descent, experience replay, and periodic updates of the target network parameters.

2.2.2. Policy-based methods

Policy-based methods optimize the policy function by adjusting its parameters to maximize the expected cumulative reward. These methods typically use gradient descent to update the policy iteratively. What follows is a list of the pros and cons of policy-based methods:

- They can handle stochastic policies, which output a probability distribution over actions rather than deterministic ones.
- They can converge to a global optimum under certain conditions, such as when the policy is continuously differentiable, and the optimization problem is convex.
- They can be computationally expensive, especially when using high-dimensional state and action spaces and complex policy representations.

Some examples of policy-based methods are REINFORCE and Deterministic Policy Gradient (DPG). REINFORCE estimates the gradient of the expected return by sampling trajectories and updating the policy parameters accordingly. The objective of REINFORCE is to maximize the expected return $\mathbb{E}[R]$ by adjusting the policy parameters θ using gradient ascent. The policy parameters are updated iteratively according to Eq. (6):

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}[R] \tag{6}$$

where α represents the learning rate, and ∇_θ denotes the gradient with respect to the policy parameters. The gradient $\nabla_\theta \mathbb{E}[R]$ is estimated using the policy gradient theorem, which decomposes the gradient into a sum of rewards weighted by the action probabilities is given in Eq. (7):

$$\nabla_\theta \mathbb{E}[R] = \mathbb{E}\left[\left(\sum_{t=0}^{T} \nabla_\theta \log \pi(a_t|s_t)\right)\left(\sum_{t=0}^{T} R_t\right)\right] \tag{7}$$

where T represents the length of the trajectory, $\pi(a_t|s_t)$ denotes the probability of selecting action a_t given state s_t, and R_t signifies the reward at time step t.

DPG aims to learn a deterministic policy directly without incorporating exploration or stochasticity. DPG updates the policy parameters using gradient ascent, guided by the estimated expected return. The policy gradient update in DPG is expressed in Eq. (8):

$$\nabla_{\theta^\mu} J \approx \hat{\mathbb{E}}\left[\nabla_a Q^\mu(s,a)|_{s=s_t,a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s)|_{s_t}\right] \tag{8}$$

where θ^μ represents the policy parameters, J is the expected return, $Q^\mu(s,a)$ is the action-value function estimating the expected return for taking action a in state s under the policy μ, and $\mu(s)$ is the deterministic policy that maps states to actions.

2.2.3. Actor–critic methods

Actor–critic methods combine value-based and policy-based methods by using separate networks to estimate the state–action value and the policy function. The actor–critic method learns to improve the policy by using the state–action value function as a baseline. What follows is a list of the pros and cons of actor–critic methods:

- They can converge faster than policy-based or value-based methods because they simultaneously learn the policy and value functions.
- They can incorporate prior knowledge of the environment by initializing the value function with a pre-existing estimate or using transfer learning techniques.
- It can be difficult to tune hyperparameters, such as the learning rate and the discount factor, which can affect the stability and convergence of the algorithm.
- The actor can overfit the current policy if the hyperparameters are not tuned correctly.

Some examples of policy-based methods are Advantage-Actor–Critic (A2C) and Proximal Policy Optimization (PPO). A2C combines the benefits of policy-based and value-based approaches by employing separate networks: an actor-network that proposes actions and a critic network that assesses state–action values. The advantage function, denoted as $A(s,a)$, is computed as the difference between the estimated value and the expected value for a given state–action pair, it is given in Eq. (9):

$$A(s,a) = Q(s,a) - V(s) \tag{9}$$

PPO iteratively improves a policy by maximizing a surrogate objective function while adhering to a constraint. The PPO objective function includes a ratio and clipping term to ensure a stable policy update. The objective function of PPO given in Eq. (10) is designed to balance between exploration and exploitation:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{10}$$

where θ represents the policy parameters, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between $\pi_\theta(a_t \mid s_t)$ which is the probability of taking action a_t in the state s_t under the policy parameterized by θ and $\pi_{\theta_{\text{old}}}(a_t|s_t)$ which is the probability of taking the same action under the previous policy, $\hat{A}_t$ is the advantage function that measures the advantage of taking action a_t in the state s_t compared to the previous policy, and ϵ is a hyperparameter controlling the policy update magnitude.

2.3. DRL hyperparameters

The hyperparameters mentioned earlier are a set of parameters that define the learning algorithm and that need to be specified by the developer. They have a big impact on the performance and convergence of the method. Here are some examples of hyperparameters:

- Learning rate: it controls the step size of the updates to the model during learning. Increasing the learning rate can help in converging faster, but it may also lead to instability and oscillations.
- Discount factor: it determines the importance of future rewards. A higher discount factor favors future rewards more than immediate rewards. This parameter can have a big impact on finding the optimal value. If badly tuned, it will converge to a local optimum instead of finding the global optimum.
- Exploration rate: it determines how much the agent should explore the action space instead of exploiting the current values that maximize the reward. A high exploration rate can help the agent better fine-tune the policy but can also lead to inefficient learning and slow convergence.
- Neural network architecture: in deep reinforcement learning, the neural network architecture can significantly affect the performance of the learning algorithm. The number of layers, the number of units per layer, and the activation functions are examples of hyperparameters that can be tuned.
- Replay buffer size: in deep reinforcement learning, a replay buffer is a memory that stores the experiences of the agent in the form of transitions, which consist of the current state, action taken, next state, and reward received. The size of the replay buffer is a hyperparameter that can affect the sample efficiency, the memory requirements, the learning stability, and the convergence of the algorithm.
- Batch size: it determines the number of experiences sampled from the replay buffer at each iteration of training. Large batch size can lead to more stable learning but requires more memory and computational capacities.

In the next section, we will mention the type of method used, the way it was trained and evaluated, and the way the hyperparameters were tuned.

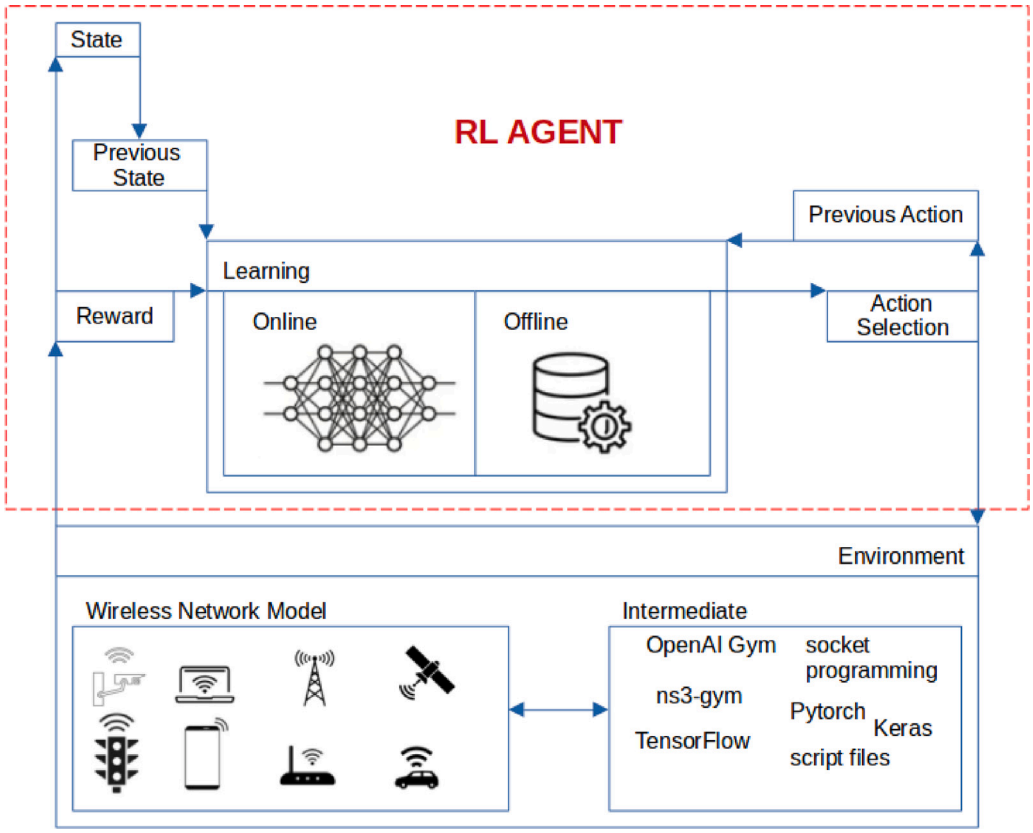


Fig. 1. The illustration of the paper.

3. Network simulators serving RL-based proposals

For real-world use, the choice of a learning model to solve a particular problem cannot be decided based on predictive performance measures alone. Limitations on computing, energy resources, and response time requirements affect the software technologies used to extract and store data, train ML models, and make predictions. In this section, we will go through the most recent efforts of using network simulators in developing techniques based on Learning for wireless networks. A schematic illustration of the paper is given in Fig. 1.

Network simulators offer unique strengths and weaknesses, such as modeling capabilities, ease of use, computational resources, and cost, that researchers and developers should consider when choosing a simulator for their specific application. We grouped the studies into 5 categories according to the network simulator used, namely, Objective Modular Network Testbed in C++ (OMNeT++), Network Simulator 3 (NS-3), Colosseum, and Matrix Laboratory (MATLAB). The 5th category includes works using other less widely used simulation tools.

In what follows, we investigate the use of each of these simulators in a separate section and end this part with a section that groups study done on other simulation platforms. At the beginning of each section, we summarize the respective strengths and weaknesses of the main network simulator. At the end of each section, we summarized the studies in a table that highlights the RL method used, the learning model, whether the simulator is used in the training or evaluation process, and if the hyperparameters of the RL model are set manually or automatically. It is extremely important not to overlook that each network simulator has its own unique features, advantages, and limitations. The choice of a simulator depends on the specific needs and requirements of the research or project at hand.

3.1. RL proposals using OMNeT++

OMNeT++ is a widely used discrete event network simulator for modeling and analyzing complex communication systems. The strengths of OMNeT++ can be summarized in the following. It allows to simulation of large-scale networks with complex communication protocols. It is flexible and allows users to specify the details of the simulated scenarios. It is open-source and extensible. On the other hand, OMNeT++ is not easy to use properly, it has limited up-to-date documentation. In addition, it has a limited graphical user interface and a limited debugging tool which make it difficult to locate and correct errors.

Authors in [27] analyzed a driver's behavior o provide individualized aid and to inform adjacent vehicles in the event of an emergency. A Refined Asynchronous Advantage Actor Critic (RA3C) algorithm is proposed. Edge-assisted Road Side Units (ERSUs) use Multi-Criterion-Based Hierarchical Correlation Clustering (MCB-HCC) to group vehicles in the road environment. A RA3C algorithm is used to analyze the driver behavior of the above-mentioned vehicles for a variety of drivers, vehicles, and environmental conditions. In the event of an emergency, alert messages are sent to surrounding vehicles of the vehicle posing a risk, which is accomplished through fog nodes using a Jellyfish Search Optimization (JSO) algorithm. The proposed Driver Behavior Analysis and Personalized Assistance (DBA-PA) model is experimented on by integrating OMNeT++ and SUMO simulation tools; SUMO acts as a traffic simulator, and OMNeT++ acts as a network simulator.

Stating that cooperative driving of Autonomous Vehicles (AVs) epitomizes the next generation in automotive technology by lowering accident risks, travel times, expenses, energy, and fuel consumption, authors in [28] present a hybrid Deep RL (DRL) and Genetic algorithm for Smart-Platooning (DRG-SP). The DQN-based Smart-Platooning incorporates Genetic Algorithm to regulate policy in accordance with the environment, resulting in increased performance. The simulation

tools such as PLEXE and Simulation of Urban Mobility (SUMO) are utilized to ensure the stability of the system and performance of the presented Smart-Platooning approach. PLEXE is a platooning simulator that integrates network simulator OMNeT++/Veins and traffic simulator SUMO. The DRG-SP algorithm is applied to the data set generated by PLEXE-SUMO using OMNeT++ and Veins.

In [29] an adaptive FD (Full Dublex) DRL algorithm (AFD-DRL) is proposed for the adaptation of vehicle users to previously unknown situations. A DQN is deployed to learn the unknown and fast-changing environment at the same time. An adaptive strategy that works in either FD or DRL mode is intended to set aside resources for broadcasting. Authors present a Veins-based Vehicular Adhoc Networks simulator that combines the wireless network simulator OMNeT++ and the traffic simulator SUMO. The simulation region is Strand, London, United Kingdom, where traffic lights and numerous lanes are both taken into account and created from Open Street Map (OSM) and turned into a SUMO network.

Stating that the Internet of Vehicles, known as IoV, is an area that provides support to vehicles with the help of internet-assisted communication, thanks to the access of more than one Radio Access Network (RAN), 5G makes connectivity ready anywhere. Authors in [30] focus on the integration of Dedicated Short Range Communications (DSRC), Long-Term Evolution (LTE), and Millimeter Wave (mmWave) in the IoV. This is combined with Handover decision-making algorithms, Network Selection, and Routing. Handover decisions are based on dynamic Q-learning algorithms in which the entropy function is utilized to forecast the threshold depending on the environment's characteristics. The Fuzzy Convolution Neural Network (FCNN) is used in the network selection process. The proposed work is done on OMNeT++ combined with SUMO.

As a result of the diversity in network technologies, it is thought and foreseen in [31] that the single technology used in the IoV is not able to fully meet the requirements of vehicular communication. Therefore, it is inevitable to work with multiple RANs to meet the requirements of safe vehicle applications. In order to propose a solution for vertical migration when working with multiple RANs, authors use a dynamic Q-learning algorithm to validate the need for handover. This is followed by appropriate network selection using a fuzzy convolution neural network. In addition, a modified jellyfish optimization algorithm is proposed to find the shortest paths by forming Vehicle-to-Vehicle (V2V) pairs. The algorithms are evaluated using OMNeT++.

Since mobile networks are characterized by a high degree of relative mobility, authors in [32] state that existing routing protocols fail to adapt their decision-making to the specified network topology dynamics. To address these challenges, they present Predictive Ad-hoc Routing fueled by RL and Trajectory knowledge (PARRoT). It is an ML-enabled routing protocol that uses mobility control information to integrate awareness about how mobile agents may operate in the coming into the routing procedure. The presented routing technique is based on Q-learning, and the performance is assessed using OMNeT++.

As the number of vehicles increases in V2V, authors in [33] claim that the channel congestion problem may arise due to the direct increase in the communication messages, and this situation adversely affects the channel performance. Although this problem can be mitigated by adjusting the data rate appropriately, it is not a preferred method because of the use of various modulation techniques, which can compromise the solution's durability in the face of unfavorable channel circumstances. In a non-cooperative scheme, authors use an analytical model to balance data rate and transmission power. They train a Deep Neural Network (DNN) in particular to properly adjust parameters for each vehicle without relying on extra data from neighbors or necessitating the deployment of additional road infrastructure. The suggested technique, known as Neural Network for Data Rate and Transmission Power (NNDP), regulates overall channel congestion while ensuring a specific transmission range with the highest achievable data rate. They demonstrate that employing a DRL approach to train a single agent is an appropriate strategy for jointly adapting data rate and transmission power and, therefore, effectively adjusting congestion levels. The training process of the DNN model is done in Python using different DRL algorithms based on Proximal Policy Optimization (PPO) and Soft Actor–Critic (SAC). In OMNeT++, many road situations are simulated in which each car broadcasts its state through the socket and the DNN model determines the appropriate transmission power and data rate.

As stated in [34], Wi-Fi networks are extremely important in time- and mission-critical applications as they guarantee the necessary widespread connectivity, especially in Industrial IoT scenarios. Moreover, the Rate Adaptation (RA) method is used to implement the multi-rate Wi-Fi capability and has shown that it is effective in improving reliability and timeliness. The proposed RL-based RA algorithm based on Q-learning considers channel behavior through the perceived SNR level. This value is used by the MDP to properly adjust the agent's states, actions, and rewards. The proposed assessment is carried out using an OMNeT++ simulation model.

By stating that the Concurrent Multi-path Transfer extension for Stream Control Transmission Protocol (CMT-SCTP), which enables multi-path and independent data streams, may help to tackle the problem of high-quality data transmission; authors in [35] propose a Q-learning based CMT-SCTP (QCMT) scheduling method. This method takes into account the path's multi-dimensional properties. OMNeT++ is used for evaluating and training QCMT with INET framework. Choosing a suitable path is done in the training process. After the completion of training, fewer packets are transferred to other paths.

In [36], it is stated that when collective intelligence is used through decentralized cooperative systems, a system called Autonomous Intersection Management (AIM) may be able to improve vehicular flow and eliminate accidents by controlling all AVs crossing urban intersections while taking into account the state of all vehicles and users. Authors examine the impact of the communication network on AIMs' decentralized control of AVs, and introduce AIM5LA, an AIM for 5G Latency-Aware. To speed up training and optimize learned information, AIM5LA uses advanced techniques and algorithms such as Twin Delayed Deep Deterministic Policy Gradients (TD3), Prioritized Experience Replay (PER), and Curriculum Learning to derive its control policy from DRL. To make a wireless communication channel and protocol simulation, the 5G simulator Simu5G is used, along with OMNeT++ and INET framework. Several scenarios are created to assess the performance of AIM5LA and the proposed messaging protocol. The first scenario is used to teach AIM5LA how to develop an advanced control strategy that takes wireless communication latency into account. The second scenario is designed to showcase AIM5LA's performance in a scenario that the algorithm had never experienced previously during training, demonstrating that the system can discover robust control even when faced with new traffic and latency scenarios.

According to authors in [37], traditional routing techniques have faced a significant problem in providing Quality of Service (QoS) as a result of the widespread use of Wireless Mesh Networks (WMN) with exponential traffic demand. They propose a QoS-assured intelligent routing system for WMNs with high traffic loads that uses RL to improve performance, namely QoS-Guaranteed Intelligent Routing (QGIR). They create a reward mechanism for the Q-Learning algorithm to find the best route for the maximum packet delivery ratio. The simulation results of the algorithm is evaluated using OMNeT++ and INET framework.

We summarize in Table 1 RL-based techniques proposed for wireless network protocols and used OMNeT++ in the process.

3.2. RL proposals using NS-3

NS3 is also a very widely used discrete event network simulator for modeling and analyzing communication protocols. The strengths of NS3 can be summarized as follows. It is realistic, it accurately models various aspects of network communication, including wireless channel

Table 1
Summary of RL proposals using OMNeT++.

Cite	Methods	Learning model	Training	Evaluation	Hyperparameter
[27]	Actor–Critic	A3C	Offline Simulation	Online Simulation	Manual
[28]	Value-based	DQN	Online Simulation	Online Simulation	Automatic
[29]	Value-based	DQN	Online Simulation	Online Simulation	Manual
[30]	Value-based	Q-Learning	Offline Simulation	Offline Simulation	Automatic
[31]	Value-based	Q-Learning	Online Simulation	Online Simulation	Automatic
[32]	Value-based	Q-Learning	Online Simulation	Online Simulation	Manual
[33]	Actor–Critic	SAC PPO	Offline Simulation	Online Simulation	Manual
[34]	Value-based	Q-Learning	Offline Simulation	Offline Simulation	Automatic
[38]	Value-based	Q-Learning	Offline Simulation	Offline Simulation	Automatic
[36]	Value-based Policy-based	PER TD3	Offline Simulation	Online Simulation	Automatic
[37]	Value-based	Q-Learning	Offline Simulation	Offline Simulation	Manual

behavior. Similarly to OMNeT++, it is open source and extensible. In addition, NS3 has a very large user community which helps in providing support. On the other hand, and similarly to OMNeT++, NS3 is difficult to use properly, it has a limited graphical user interface, and limited up-to-date documentation. In addition, NS3 is known to be very resource consuming when dealing with large-scale mobile topologies.

NS-3 is an open-source network simulator that is free to use and supports a large number of networking standards. It is perfect for reproducing intricate and vast networks thanks to its inherent flexibility and scalability. On the other hand, NS-3 has a steep learning curve and few graphical user interface (GUI) capabilities, which might make it challenging for beginners to get started. In addition, the high processing demands of large-scale simulations might be a hindrance to some research goals.

In [39], authors propose a DRL approach for the automatic Rate Adaptation. By using DRL techniques the algorithm that they presented, DARA (Data-driven Algorithm for Rate Adaptation), can learn the best Wi-Fi Modulation and Coding Scheme to utilize in a given time interval by calculating the average SNR of the frames received in the previous time interval at the transmitter node. The NS-3 simulator and NS3-gym, a framework that allows the building of an OpenAI Gym RL interface for the NS-3 simulator, are used to train and evaluate DARA. The DARA agent is also implemented in Python and uses the TensorFlow package. The performance of DARA is compared to different RA algorithms. DQN is proposed to deal with instances where the number of possible observations and actions is limited.

Authors in [40], present an RL agent based on Q-learning to manage the data transmission rates of nodes in wireless networks based on carrier sensing multiple access with collision avoidance (CSMA/CA). The NS3-gym framework is used to simulate RL agent, which responds to changes in wireless networks and modifies the modulation and coding scheme levels of data packets to adaptively manage the data-sending speeds of nodes in wireless networks.

To implement AI (Artificial Intelligence) algorithms for wireless network optimization, NS-3 is used in [41]. Besides the ability to train and evaluate the algorithms in a simulated but realistic and controlled environment, the authors illustrate the framework's correctness and technical soundness in a test scenario where network performance is managed using Predictive QoS. The RL agent is trained in accordance with the Double Q-Learning (DQL) algorithm. Also, in order to implement a geometry-based channel model for V2X, a detailed representation of the area is obtained via OSM and then converted into a SUMO network file.

For routing protocols to adapt their decision-making processes in a timely manner to the variability of real network conditions, and to provide reliable and efficient data distribution mechanisms, in [42], they demonstrate the NS-3 integration of PARRoT. Through the Q-learning algorithm, the node keeps track of its neighbors, storing their sequence number, anticipated position, and last contact timestamp. The implementation of the routing protocol on NS-3 is presented, and pointed out structural differences to OMNeT++ in terms of the migration process.

In [43], an algorithm named Neuro-DCF is proposed, which adopts an experience-driven approach and train carrier sense multiple access based wireless Medium Access Control (MAC) by using DRL. To support a stable training method for distributed execution, authors adopt a multi-agent RL framework. To support a unified training method for embracing various interference patterns and configurations, they introduce a new graph neural network. The PPO is used and modified for the baseline RL algorithm for updating the actors in the cooperative multi-agent RL problem. NS-3 is used for the implementation of the wireless protocol. The NS3-gym framework is also used for connecting the simulator and Python-based OpenAI Gym.

In case of needs of communications with high bandwidth, reliability, and minimal latency, stating the necessity of AI-powered algorithms for resource block allocation, authors in [44] propose an Advantage-Actor–Critic (A2C) Learning based scheduler to allocate resource blocks in a re-configurable wireless network. They formulate the number of resource blocks (RBs) and their location in the RBs' map as a Markov Decision Process (MDP). They solve the problem using an A2C. The NS3-gym is used for implementing the proposed algorithms and connecting the simulator with OpenAI Gym. For the neural networks, Pytorch is used.

By stating that the proper setting of Contention Window (CW) values has a significant impact on the efficiency of Wi-Fi networks and 802.11 networks' typical technique is not extensible enough to keep throughput consistent as the number of stations grows, in [45], authors propose a new method of CW control, called Centralized Contention window Optimization with DRL (CCOD). DQN and Deep Deterministic Policy Gradient (DDPG) are used. CCOD is implemented in NS3-gym.

PyTorch and TensorFlow are used to demonstrate the independence of agent implementation. They obtained a trained agent as a consequence of the learning process, which can be directly placed in an 802.11ax application.

Authors in [46], propose a DRL-based load-balancing approach for cellular networks as a way to solve of LTE traffic problems. The proposed model employs a Double DQN (DDQN) to learn how to alter Cell Individual Offset parameters and eNodeBs' transmission power in order to improve overall network load distribution. The simulation is applied using NS-3. The RL agent uses DDQN for learning the best control of the environment and implemented using Python and TensorFlow. Also, NS3-gym is used as an interface.

In [47], authors provide an MDP method to jointly maximize downlink transmitting energy and cell-specific offsets in cellular networks. They use the TD3 technique to discover how to get the most out of the suggested reward function utilizing the NS-3 and SUMO simulators in conjunction with the cellular network. Python is used to build the agent, which is based on the Open AI Gym. NS3-gym serves as the interface between the NS3-based environment and the agent.

In [48], authors offer a Clipped DQL-based load balancing strategy that considers resource block use, latency, and the Channel Quality Indicator in order to maximize basic network usage or overall network throughput. In the proposed approach, the agent is based on DDQN and the simulations are conducted on NS-3, for the aim of the interface between the simulator and the agent OpenAI Gym is used.

In [49], to achieve high QoS in Long Rage (LoRa) communications, authors present transmission policy enforcement and Multi-hop routing for QoS-aware LoRa networks (MQ-LoRa). Transmission policy enforcement uses SAC learning, and multi-hub routing determines mayfly and shuffled shepherd optimization to schedule routes according to the fitness criteria. The efficiency of the algorithm is shown on NS-3.

Authors in [50] focus on how deep Q-learning technologies can be employed in an Active Queue Management (AQM) algorithm to cut down on the amount of time spent queuing while still ensuring optimum connection utilization. The suggested method, entitled RL-AQM and implemented in NS-3 using its OpenAI Gym extension, has the advantage of being less prone to bad parameterization and being able to respond to changing network circumstances automatically. The queue delay control is done by a DQN-based agent that gets knowledge regarding the congestion connection and the queue on a regular basis and determines a plan of steps to be taken that alters the drop possibility of the queuing discipline.

In [51], carrier aggregation of LTE and 5G networks is noted as an important enabling technology to provide higher rates to their users. The authors state that the increased transmission rate reduces energy consumption due to users constantly checking the active component carriers' control channel to see if data transfer is ongoing. A study is conducted on the enable-disable procedure at the MAC layer of the LTE/5G network. They offer an RL-based method to support efficient usage of energy, it automatically enables and disables secondary component carriers based on user traffic profiles. Using Q-learning, the proposed technique seeks to foresee when data will arrive and define the secondary component carriers to activate for each user.

Authors in [52] offers an RL-based framework for IEEE 802.11 standards wireless channel access in massive IoT. The proposed mechanism, based on Q-learning, proposes using a wireless environment gathered data-set of channel collision possibility as assessed in the real world to identify efficient use of resources in massive IoT for impending 6G wireless communications. The simulations are carried out on NS-3.

In [53], vehicle mobility and high dynamic network architecture are investigated. Authors RLSS, an RL-based data Source Selection scheme for effective high-definition map circulation in vehicular named data networking (NDN) scenarios. It aims to find the best data source ion the strength of map data demands. The RL method for training the model based on Q-learning is the DDQN methodology. They use the approach to train a neural network as an agent to choose data sources, which can work online following offline training based on prior behavior. RLSS is implemented in NS-3 with the help of the ndnSIM, SUMO, and Gym tools.

Authors in [54] consider that a self-managed network is necessary to modify the high cost, time-consuming and manual management proned to errors. They propose a DRL-based cellular network management system. The proposed system is aimed at the integrity of the network and adapting to temporary changes. With the presented approach, a network administrator can maximize the average total efficiency of the network while attempting to reduce the amount of energy used and the number of banned users. To solve this issue, in a layered approach, the model uses two RL algorithms. The DDQN is utilized for discrete action spaces, and the TD3, is utilized for continuous action spaces. NS-3 is used to model the environment, which is represented by a cellular network. Python is used to imitate the agent. The NS3-gym interface is used to create a connection between the agent and the environment.

Authors in [55] focus on an RL strategy for managing intelligent traffic signal scheduling. They employ a method that receives data from the vehicular network and controls the different stages of traffic lights based on the current status of the traffic flow in real-time. The TraCI API is used to create an RL training environment involved in real circumstances. To train the RL agent, which is based on DDQN, SUMO, and NS-3, simulators are used together and provided by the OpenAI Gym.

Authors in [56] use a DRL approach capable of handling large state spaces in order to manage task offloading in IoT systems. The agent, based on DQN, is responsible for translating the condition of the IoT system resources to the best execution options. In the model they proposed, NS-3 is used for representing the environment, while the OpenAI Gym unifies their interface, and DQNs are built by using TensorFlow.

By expressing as TCP is unable to discern the cause of packet loss, traditional TCP end-to-end congestion control algorithms cannot be directly applied to heterogeneous wireless networks. authors in [57] developed a model that anticipates the best congestion window by dynamically interacting with the surroundings. They present an RL model that uses the A2C technique and TD learning to dynamically alter the congestion window. The NS-3 simulation tool is used for implementing the proposed heterogeneous wireless network, and OpenAI Gym is used for the agent to interact with the environment.

As a solution to choose various networks in various user scenarios and to remove limits to the network access' usefulness, an algorithm based on DDQN is suggested to optimize the network configuration in [38]. A Self-selection Protocol algorithm for Wireless Networks based on DDQN is proposed to increase the level of stability of heterogeneous wireless networks and network throughput and preserve a rather more steady state in a dynamic network that is continually changing. By combining the RL in the MDP process, the DDQN algorithm is selected to solve the problem of possible DRL exaggeration. The DDQN algorithm is built in Python using OpenAI Gym and the network environment design through NS-3.

Table 2 provides a summary of RL-based proposals that were either trained or evaluated using NS-3.

3.3. RL proposals using Colosseum

Colosseum is more of a wireless network emulator than a simulator. But since is relatively new and starting to become famous, we decided to include in the survey. It is designed to experiment with wireless networking protocols in a reproducible manner. The strengths of the Colosseum can be summarized in what follows. It is realistic and allows to the creation of large-scale networks. The experiments can be reproduced in a controlled manner which allows for to enhancement of the credibility of the results. It offers a user-friendly web-based interface for designing, deploying, monitoring, and analyzing experiments. On the other hand, Colosseum provides a fixed set of hardware and software

Table 2
Summary of RL proposals using NS-3.

Cite	Methods	Learning model	Training	Evaluation	Hyperparameter
[39]	Value-based	DQN	Offline Simulation	Offline Simulation	Automatic
[40]	Value-based	Q-Learning	Offline Simulation	Offline Simulation	Manual
[41]	Value-based	DQL	Offline Simulation	Offline Simulation	Manual
[42]	Value-based	Q-Learning	Offline Simulation	Offline Simulation	Manual
[43]	Actor–Critic	PPO	Online Simulation	Online Simulation	Manual
[44]	Actor–Critic	A2C	Offline Simulation	Offline Simulation	Manual
[45]	Value-based Policy-based	DQN DDPG	Online Simulation	Online Simulation	Manual
[46]	Value-based	DDQN	Offline Simulation	Offline Simulation	Manual
[47]	Actor–Critic	TD3	Online Simulation	Online Simulation	Manual
[48]	Value-based	DDQN	Offline Simulation	Offline Simulation	Manual
[49]	Actor–Critic	SAC	Offline Simulation	Offline Simulation	Manual
[50]	Value-based	DQN	Offline Simulation	Offline Simulation	Automatic
[51]	Value-based	Q-Learning	Online Simulation	Online Simulation	Manual
[52]	Value-based	Q-Learning	Offline Simulation	Offline Simulation	Automatic
[53]	Value-based	DDQN	Online Simulation	Online Simulation	Manual
[54]	Value-based Value-Based	DDQN TD	Offline Simulation	Offline Simulation	Automatic
[55]	Value-based	DDQN	Online Simulation	Online Simulation	Automatic
[56]	Value-based	DQN	Online Simulation	Online Simulation	Automatic
[57]	Actor–Critic	A2C	Offline Simulation	Online Simulation	Manual
[38]	Value-based	DDQN	Online Simulation	Online Simulation	Manual

models. It does not support mobility. It is used under limited access and time periods through a reservation process which can be challenging for extensive use and experiments for everyone.

Authors in [58] highlight the benefits of using RL in an Open Radio Access Network (O-RAN) to make decisions based on the state of the network and optimize network performance. The authors discuss several RL algorithms that can be used in O-RAN, including Q-learning and policy gradient methods, and provide examples on how RL can be used to optimize network resource allocation, network policies, and network configuration. A set of DRL agents are trained offline to optimize key performance metrics for different network slices via data-driven closed-control loops.

Authors demonstrate how the Colosseum platform can be used to build a cellular infrastructure managed by an O-RAN in a given area in [59]. To do this, closed-loop control is used, which is data-driven and enabled in near real-time by xApps embedded in RAN Smart Controllers. Three slices of available spectrum are split and base stations use DRL agents deployed as xApps to process the configuration of each slice. These agents transmit control commands to the base stations via the O-RAN system after periodically receiving data reports from the base stations. Results show that network performance is maximized with this method.

Authors in [60] present xApps on an open-source platform for RAN experiments, as an intelligent closed-loop control mechanism for RANs. RAN controllers on which xApps is installed collect data and increase network performance using machine learning methods. xApps are trained using a reinforcement learning strategy that allows them to adapt to changing network conditions and choose the best control strategies. Results show that xApps can effectively manage RAN and optimize network performance in real-time.

In [61] the development of machine learning-based xApps for closed-loop control in O-RAN architectures is discussed. The authors profile the RAN performance during the exploration phase of DRL and after the training phase. On an offline data set employing traffic based on network slices, DRL agents have been trained. They demonstrate how an additional online training step adjusts a pre-trained model to the deployment-specific parameters, fine-tuning its weights at the expense of a temporary degradation in performance during the online exploration phase.

The QCell framework, developed particularly to leverage deep Q-learning techniques for real-time self-optimization of network parameters, is introduced by the authors of [62]. The authors have shown the QCell architecture to be successful in enhancing network performance metrics like throughput and latency through simulations and tests. They

Table 3
Summary of RL proposals using Colosseum.

Cite	Methods	Learning model	Training	Evaluation	Hyperparameter
[58]	Actor–Critic	PPO	Offline Simulation	Offline Simulation	Manual
[59]	Value-based	DQN	Offline Simulation	Offline Simulation	Manual
[60]	Value-based	DQN	Offline Online Simulation	Offline Simulation	Manual
[61]	Actor–Critic	PPO	Online Offline Simulation	Offline Simulation	Manual
[62]	Value-based	DQN	Offline Simulation	Offline Simulation	Manual
[63]	Value-based	DQN	Offline Simulation	Offline Simulation	Manual
[64]	Value-based	Q-Learning	Offline Simulation	Offline Simulation	Manual

also go over the QCell framework's scalability and resilience, which make it an attractive option for usage in expansive 5G networks. They describe training DRL agents on an offline data set and an additional online training step to adapt a pre-trained model to deployment-specific parameters.

In [63], authors offer a DQL experiment for dynamic spectrum sharing. A wireless network testbed is built using a software-defined radio architecture with numerous primary users and one secondary user. While secondary users utilize DQL to dynamically choose channels to broadcast based on available spectrum, primary users employ a fixed spectrum allotment. This proves that the DQL-based strategy can efficiently share the spectrum with primary users while increasing secondary user throughput. The experiment's findings demonstrate that DQL has the potential to be used in dynamic spectrum-sharing applications and that it can result in more effective spectrum usage.

In a mixed cooperative/competitive context in [64], the study provides a RL strategy for dynamic spectrum access. The authors suggest a multi-agent RL framework in which each agent is in charge of deciding on a transmission channel in accordance with its own goals and those of other agents. Agents employ Q-learning to discover the best channel selection strategy, which considers both the agents' individual and group objectives. The findings demonstrate that the RL-based method outperforms conventional dynamic spectrum access schemes in terms of throughput, fairness, and network usage. It is also capable of allocating spectrum in mixed cooperative/competitive contexts.

Although the use of the Colosseum is not as large as the previously studied open-source simulators, Table 3 summarizes the RL-based wireless network concepts that used the Colosseum platform throughout the training and assessment stages.

3.4. RL proposals using MATLAB

Although network simulators are the preferred option for testing wireless protocols, it is important to note that RL methods in wireless network technologies are also frequently numerically tested using MATLAB. Even though MATLAB is not a network simulator, it is nevertheless widely used for evaluating RL models for wireless networks. MATLAB is a numerical computing environment that models and simulates various systems, including communication systems. The strengths of MATLAB can be summarized in what follows. It has a user-friendly interface and a large library of functions allowing it to simulate communication systems. It uses high-level programming that allows it to easily encode complex mathematical operations. It is flexible and can be integrated with other tools such as Simulink and C++ for designing and testing communication systems. On the other hand, similarly to previously studied tools, it is resource-consuming when dealing with large-scale scenarios with limited support for parallel processing. Moreover, as proprietary software, MATLAB can be expensive for some users.

Authors in [65] propose a QoE-driven edge caching method for the IoV based on DRL. According to the authors, cache refresh mechanisms that are presently being used, such as partial file pre-caching and assessing content popularity and user interest, are ineffective. As vehicles can communicate with each other and with cloud servers while driving. Authors work on cache update optimization based on file download rates from RSU and cloud servers. To do so, a user interest model based on class is determined. Based on the proposed class-based user interest model, an effectiveness of experience QoE-driven RSU cache model is constructed. A DRL method is intended to effectively address the QoE-driven RSU cache update issue. In the A2C method, the actor-network calculates the next state's behavior, while the critic network calculates the action value function to appraise the state using a Temporal Difference (TD) learning technique. The results of the simulations validate the proposed algorithm's effectiveness.

Due to channel interference and wireless spectrum scarcity, it is critical to efficiently multiplex channel resources in wireless networks. In [66], the problem of Wireless channel resource increasing in vehicle networks is investigated. To provide a more precise target evaluation for action selection, a decentralized MDP and a multi-agent distributed channel multiplexing system based on Prioritized DDQN are suggested. This helps in resolving the problem of overestimation in action evaluation. After training, each V2V Agent implements a learning framework that is dynamically changed by network environmental rewards to maximize wireless resource utilization. The experimental results are obtained from MATLAB.

In [67], authors present a QoS-maximizing intelligent RAN slicing technique. The proposed strategy is made up of an upper-level controller and a lower-level controller. The upper-level controller changes the slice configuration to enhance QoS performance. Physical resource blocks and power allocation to working UEs for each network slice are quite well planned by the lower-level controller. Then, based on multi-timescale MDP, authors propose a hierarchical DRL framework for learning the optimal RAN slicing control strategy, which is an integration of modified DDPG and DDQN algorithms. Simulation results, done with Python and MATLAB, demonstrate that the suggested approach exhibits steady convergence control performance.

Authors in [68] concentrate on enhanced mobile broadband services, which enable high-data-rate communication and low-latency communications systems that are ultra-reliable in order to ensure real-time connectivity to applications that require a minimal delay. Because the computation of the solution necessitates the employment of a channel and/or traffic prediction method, which is not practical in

most real-world settings. To address these issues, an alternate resolution approach based on DRL is proposed, which computes a suitable spectrum allocation policy using a model-free approach to the problem. They improve the training performance of the DQN agent by using an action masking technique that enables the agent to reject activities that do not add to the total reward. The achieved normalized reward is used to compare agent learning performance between the two network scenarios under consideration. When the agent is used in a mobile network scenario, it is discovered that it converges in a lesser number of episodes. MATLAB is used to develop both the agent and the network environment.

In [69], authors aim to reduce energy consumption at the network level while meeting various QoS requirements in 6G wireless networks. They present a new Multi-Agent Distributed Q-Learning based Outage-aware Cell Breathing (MAQ-OCB) framework for jointly optimizing energy efficiency and user outage. In this architecture, each agent examines the state information of its neighbor's small cell base stations, and the reward is divided among all base stations. The algorithm uses a decayed epsilon greedy approach to allow the agent to learn about network surroundings and explore more diverse states. The training and simulation setups of the proposed algorithm are implemented in MATLAB.

To provide a solution to dynamic route establishment in dynamic wireless networks where existing non-adaptive routing algorithms are insufficient, in [70], authors propose RL-based self-adaptable Q-routing in the cognitive radio networks with various network topology. The goal is to achieve the best secondary throughput performance. The simulations in different network topologies are conducted on MATLAB to see reducing the run-time complexities of the Q-routing.

The authors of study the dynamic power allocation scheme in the network citetan2022resource using Q-learning and DQN in order to maximize the average throughput of Fog-RAN (F-RAN) architecture with hybrid energy sources and satisfy the constraints of SNR, available bandwidth, and energy harvesting. MATLAB is utilized for numerical simulation to verify the efficiency of the suggested resource allocation technique.

Authors in [71] improve Q-Learning in MIxing network in a complex nonlinear way (QMIX) which is a well-known multi-agent RL algorithm based on DQN. By introducing an additional specific Q-value for each agent in the mixing network in addition to the main total Q-value, they make QLBT stable. The proposed method called QMIX-advanced Listen-Before-Talk (QLBT) uses Centralized Training with Decentralized Execution (CTDE) framework to trade on all agents' information gathered throughout the training. It provides that each agent can deduce the best channel access behavior for itself on the basis of its surrounding's observations. To get numerical simulation results, MATLAB is used.

In [72], authors proposed a novel Multi-Agent RL-based (MARL) distributed localization scheme. First, the localization problem is formulated as a stochastic game with the goal of maximization of the adverse localization errors totals. each non-anchor node is modeled as an intelligent agent with an action space that relates to possible locations. To train the optimal strategy and increase the long-term expected value, authors adopt a MARL framework based on the conventional Q-Learning paradigm. Both in terms of localization precision and convergence rate, the proposed localization approach outperforms the game theoretic-based decentralized positioning algorithm and the decentralized positioning algorithm based on virtual dynamics when tested on MATLAB.

Table 4 provides a summary of RL-based proposals for wireless networks using MATLAB in the training and or evaluation phases.

3.5. RL proposals using other less widely used simulators

In this part, we provide a quick overview of other less widely used simulation platforms but that are also used for the development and training of learning-based algorithms for wireless networks.

By stating that Q-routing uses latency as a routing metric, which can be difficult to assess, especially in a wireless environment with a lot of noise, authors in [74] propose using a moving average to filter the latency measure in order to improve the QoS measures. They evaluate the modified Q-routing algorithm, based on Q-learning, with filtration on many ad-hoc wireless network situations, along with a mobility simulation on QualNet, derived from GloMoSim.

DeepPlace, an adaptive flow rule placement system based on DDPG, is proposed for use in Software-Defined Internet of Things (SDIoT) networks by authors in [75]. In addition to proposing a cost function for routing to establish data plane traffic flow paths, they suggest an adjustable flow rule placement way to expand overall a flow rule's number of match-fields at SDN switches. They use the MDP with a constant action space to model system functioning and develop an optimization problem to strive for the dynamism of IoT traffic flows. To evaluate the proposed DeepPlace framework, the authors use ONOS (Open Network Operating System) as an SDN controller, using the GoodNet tool to simulate a real-world network architecture.

In [76], the Federated DRL (FDRL) method is used to estimate future demands from the user and select the suitable content replacement method in a Hierarchical FDRL (HFDRL). Furthermore, the mechanism by which partner devices are stated is crucial to edge caching performance in order to do learning and collaborative caching. Edge devices are categorized in a hierarchical order. HFDRL avoids the drawbacks of either very small or very large clusters while taking advantage of both. In the HFDRL system, each edge device trains its neural network to calculate Q-value and anticipate the prominence of the items demanded by its user, which is then used to decide whether it is better to cache or remove content. Each local base station network's performance is improved separately and for the worldwide network by minimizing content storage redundancy and latency. The proposed model, HFDRL, is simulated using EdgeCloudSim in Java.

Software-Defined Networking (SDN) provides fast and dynamic deployment of network policies commonly used in enterprise and wide area networks, with a global view of the network to set continuous routes and acquire network connectivity states. To meet the requirements of various services, corresponding QoS policies must be enforced, and a balance of link utilization is also required. Reinforcement Discrete Learning-Based Service-orienTed Multi-pAth Routing (RED-STAR) is proposed in [77]. The RED-STAR mechanism based on DQN, uses the network state and service type as input values to dynamically determine which path a service must take. Custom protocols are created for obtaining network state, and a DRL-based traffic classification model is included to identify network services. Mininet is used as a network simulator. Results show that the DRL model in RED-STAR progressively provides superior reward values in various scenarios thanks to the differentiated reward scheme for each service type.

Authors in [78] stated that due to the continuous topology changes caused by the rapid mobility of UAVs (Unmanned Aerial Vehicles), they pose a serious problem of poor communication quality, the high mobility of nodes makes it difficult to predict the state of the link, so more frequent updates are required for link quality. The authors propose a Q-learning-based routing protocol that takes into account the minimum number of hops to the target, the quality of the connection, and mobility to adapt the update period to environments. To create Multi-UAV Networks, a cascading neighbor node search is first conducted to represent the link's stability and the shortest number of hops to a target. Next, they design the link stability factor with the number of hops combined with the reward function for the Q-learning mechanism by performing the Q-learning algorithm based on the information obtained by the progressive node search. They used Optimized Network Engineering Tool (OPNET) to assess the achievements of the protocol.

According to the authors in [79], setting the threshold of each priority queue prevents the transmission of low-priority packets when there are few high-priority packets and ensures the performance of latters. They propose a DQN-based transmission and back-off (DQNTB)

Table 4
Summary of RL proposals using MATLAB.

Cite	Methods	Learning model	Training	Evaluation	Hyperparameter
[65]	Actor–Critic	A2C	Online Numerical	Online Numerical	Manual
[66]	Value-based	DDQN	Online Numerical	Online Numerical	Automatic
[67]	Policy-based Actor–Critic	DDPG DDQN	Online Numerical	Online Numerical	Manual
[68]	Value-based	DQN	Offline Numerical	Online Numerical	Manual
[69]	Value-based	Q-Learning	Offline Numerical	Offline Numerical	Manual
[70]	Value-based	Q-Learning	Offline Numerical	Offline Numerical	Manual
[73]	Value-based	Q-Learning DQN	Offline Numerical	Offline Numerical	Automatic
[71]	Value-based	DQN	Online Numerical	Online Numerical	Manual
[72]	Value-based	Q-Learning	Offline Numerical	Offline Numerical	Automatic

Table 5
Summary of RL proposals using different simulation platforms.

Cite	Methods	Learning model	Training	Evaluation	Hyper-parameters	Tool
[74]	Value-based	Q-Learning	Offline	Offline	Manuel	QualNet
[75]	Policy-based	DDPG	Online	Online	Manuel	ONOS Maxinet
[76]	Value-based	Q-Learning	Offline	Offline	Automatic	EdgeCloudSim
[77]	Value-based	DQN	Online	Online	Automatic	Mininet
[78]	Value-based	Q-Learning	Offline	Offline	Automatic	OPNET
[79]	Value-based	DQN	Offline	Offline	Automatic	OPNET
[80]	Value-based Value-Based	DQN DDQN	Offline	Offline	Automatic	NetSim

method to optimize channel usage and reduce the back-off time for low-priority traffic. The simulation of the protocol, which comprises two stages, the transmission network, and the withdrawal network, is done using OPNET.

In [80], an RL-IDS (Intrusion Detection System) is proposed to enhance the strength of distributed ML-IDS in detecting internal and external Routing Protocol for Low Power Lossy Networks (RPL) intrusions. DQN and DDQN algorithms are used to find an optimum policy that results in the maximum long-term reward. The NetSim simulator is used to simulate different RPL attack scenarios and generate raw data sets.

Table 5 provides a summary of RL-based proposals for wireless networks performed on different simulation platforms.

4. Open issues and challenges

Throughout this paper, we explored the utilization of various network simulators for RL studies. Some researchers used the network simulator in the training phase of the RL agent, while others used them both for training and evaluation. Indeed, network simulators have been time-saving and scalable tools to develop and evaluate RL approaches. However, many issues and challenges should be addressed, such as: the credibility and reliability of the simulators, the fine-tuning of the hyperparameters of RL algorithms, and the generalization ability and maintenance of the deployed model. In what follows we will discuss each of these challenges.

4.1. Training environment

A RL agent must interact with an environment in its training phase. During training, the agent observes the states of the environment, takes action, and learns from the obtained rewards. In wireless networking, the environment may consist of a wireless network formed of real devices or built in a simulation tool. Our study has shown that researchers mainly use network simulators for training instead of running hardware experiments. Simulators attempt to emulate the real-world behavior of a wireless network. They are scalable and allow testing scenarios of varying field conditions by adjusting only a few lines of code. On the other hand, a wireless network simulator should be realistic and reliable.

Wireless environments are dynamic by nature. A network simulator cannot accurately model all the dynamics of an environment. The availability of various simulators makes reliability a challenge. A simulation can only be useful when the results are comparable to real experiments. For instance, authors in [81] attempt to validate the Wi-Fi MAC model of NS-3 by using different scenarios to compare their testbed measurements to the simulation results. The authors conclude that the Wi-Fi MAC model of NS-3 is a good representation of reality, although they observed some quantitative differences. Authors stress that it is not necessarily the fault of the model but may also be a flaw in the implementation of the testbeds.

Other studies, such as in [82,83], found similarities between the experimental measurements and simulation results when testing certain aspects of the physical and MAC models of OMNeT++ and NS-3.

Open-source simulators such as NS-3 [84] and OMNeT++ [85] have been accepting contributions from experts in the wireless domain over

the years. Maintainers and the community always work on adding support for new wireless technologies, features, and fixing bugs. Home-made simulators are closed-source projects, they lack the added value of open-source projects. They do not necessarily implement all the features of the wireless network layers and are less credible due to their closed-source nature.

4.2. RL parameters

RL is not a plug-and-play solution that blindly solves any kind of problem. The learning process is sensitive to a set of adjustable hyperparameters that should be fine-tuned. In addition, the reward function that drives the learning process should be engineered well to aid the algorithm in reaching the desired objective.

4.2.1. Hyperparameters tuning

Hyperparameters such as the learning rate, discount factor, batch size, and the number of hidden layers are used to tune RL algorithms. Setting the value of a hyperparameter poses a challenge as it is essential to improve the performance of the RL model. Most of the RL approaches in the literature tend to tune the hyperparameters manually. Authors in [86] tried setting different learning rate values with the objective of lowering the power consumption in green F-RAN. Manually tuning the learning rate showed that setting a high value caused reaching a local optimum while reducing it slowed the learning process but led to better results. However, manual tuning leads to reaching an optimum relative to the selected set of a certain hyperparameter.

The work in [87] analyzes the application of hyperparameters optimization techniques (i.e., Bayesian Optimization [88]) in DRL. Hyperparameters are optimized while attempting to solve an indoor wireless localization problem. When automatic tuning is employed, location prediction errors are lower than when manual tuning is applied.

Hyperparameter tuning has an enormous impact on the training time and the accuracy of the trained models. It makes DRL models more reliable for wireless environments, which are dynamic by nature.

4.2.2. Reward function definition

The reward function is one of the crucial elements that influence the quality of the RL model. RL approaches define a reward function based on the application objective. A reward function may rely on a single or multiple metrics, often of the same layer where the study lies in. However, only a few studies have considered adding metrics of other layers. The aim of adding additional metrics is to extend the observation space of the RL agent or include the metrics in the reward function. In [89], a DRL model is developed to perform resource allocation in cognitive radio networks to maximize QoE of video transmissions. When using cross-layer metrics (network and physical layers), the DRL model produced a 1.4 times better Mean Opinion Score (MOS) than using only physical layer metrics. Cross-layer metrics provide a broader view of the network, and their correlation may result in enhancing the trained model.

4.3. Generalization and maintenance of the model

Neural networks are at the core of DRL. A neural network is a complex function that maps a set of inputs to a set of outputs. Even though DRL introduced improvements in solving problems in the wireless domain, the neural network itself remains a black box. Neural networks are not deterministic and cannot be mathematically modeled. Ensuring that a trained DRL model solves a certain wireless network problem is not an easy task. One has to test all the input combinations to make sure that the model is predicting the desired actions in all situations. In wireless networks, many parameters are real numbers, such as throughput, packet loss, latency, jitter, etc. Thus, DRL solutions are only applicable with a certain degree of confidence and one cannot be sure that the DRL model can be generalized to any untested situation.

During DRL training, the values of the neural network parameters, such as the weights and biases, are adjusted based on the DRL algorithm and the reward definition. However, the adjustment of the parameters is not mathematically related to the outcome of the neural network. Thus, if any error appears after deploying the model, it cannot be dealt with by simply adjusting the neural network parameters in a certain direction. The whole training process should be repeated and validated. Thus, the maintenance of the deployed model is critical and time-consuming, especially if the model is deployed in a production environment.

5. Conclusion

In this survey, we covered most work that relied on simulation tools, mainly network simulators, in order to train, validate, evaluate, and explore RL methods designed for wireless networks. We presented a brief functional description of RL methods highlighting the main characteristics of each type of method. Also, we cited the most relevant hyperparameters that have a big impact on the results of these methods.

Many simulation platforms exist, some of them are very widely used due to their open-source and extensibility, others are more restricted but more reliable. We briefly summarized the strengths and weaknesses of the main used simulators in this area of research.

We went through the most recent research work in this domain, mainly from the past 3 years, and highlighted how the simulation tool helped the study. We grouped the studies according to the simulator used and summarized each group in a dedicated table that highlighted the type of the RL method, the learning model, the way the training and evaluation were conducted, and the manner in which the hyperparameters were chosen.

We ended the survey with a summary of the current challenges and open issues related to this domain. Mainly, the demystification of the parameters of neural networks used with DRL, the fine-tuning of the other hyperparameters, as well as the reliability of the simulation tools, and the transition to real-world validation all remain what the researchers in this domain are working hard on.

In conclusion, choosing the right simulation platform that better serves the Rl method will very much depend on the exact application and use case for which it is used. When dealing with low-level RL agents that will take real-time actions related to the physical nature of wireless networks, the use of simulation tools might not be enough. Hybrid and complementary methods that make the best of the simulation tools in terms of scalability, availability, and extensibility should be coupled with validation on real hardware with fine-tuning training to ensure the transition between the simulation environment and the real world.

CRediT authorship contribution statement

Serap Ergun: Conceptualization, Methodology, Resources, Writing – original draft, Writing – review & editing, Visualization. **Ibrahim Sammour:** Methodology, Resources, Writing – original draft, Writing – review & editing. **Gerard Chalhoub:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Gerard Chalhoub reports financial support was provided by French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25).

Data availability

No data was used for the research described in the article.

Acknowledgment

This research was funded by the French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25). All authors have read and agreed to the published version of the manuscript.

References

[1] B. Sharma, R. Saini, A. Singh, K.K. Singh, Deep reinforcement learning for wireless network, Mach. Learn. Cogn. Comput. Mob. Commun. Wirel. Netw. (2020) 51–71.
[2] M.S. Frikha, S.M. Gammar, A. Lahmadi, L. Andrey, Reinforcement and deep reinforcement learning for wireless internet of things: A survey, Comput. Commun. 178 (2021) 98–113.
[3] A. Mamadou Mamadou, J. Toussaint, G. Chalhoub, Survey on wireless networks coexistence: resource sharing in the 5G era, Mob. Netw. Appl. 25 (5) (2020) 1749–1764.
[4] N.C. Luong, D.T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, D.I. Kim, Applications of deep reinforcement learning in communications and networking: A survey, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3133–3174.
[5] B. Fayssal, A. Marwen, D. Fedoua, Network selection schemes in heterogeneous wireless networks, 2022, arXiv preprint arXiv:2201.12021.
[6] I. Shayea, M. Ergen, M.H. Azmi, S.A. Çolak, R. Nordin, Y.I. Daradkeh, Key challenges, drivers and solutions for mobility management in 5G networks: A survey, IEEE Access 8 (2020) 172534–172552.
[7] R. Pirmagomedov, D. Moltchanov, A. Samuylov, A. Orsino, J. Torsner, S. Andreev, Y. Koucheryavy, Characterizing throughput and convergence time in dynamic multi-connectivity 5G deployments, Comput. Commun. (2022).
[8] M. Rani, K. Ahuja, Interface management in multi-interface mobile communication: a technical review, Int. J. Syst. Assur. Eng. Manag. (2022) 1–13.
[9] M. Chen, U. Challita, W. Saad, C. Yin, M. Debbah, Artificial neural networks-based machine learning for wireless networks: A tutorial, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3039–3071.
[10] A. Mekrache, A. Bradai, E. Moulay, S. Dawaliby, Deep reinforcement learning techniques for vehicular networks: recent advances and future trends towards 6G, Veh. Commun. (2021) 100398.
[11] N.K.S. Nayak, B. Bhattacharyya, Machine learning-based medium access control protocol for heterogeneous wireless networks: A review, in: 2021 Innovations in Power and Advanced Computing Technologies (i-PACT), IEEE, 2021, pp. 1–6.
[12] E.S. Ali, M.K. Hasan, R. Hassan, R.A. Saeed, M.B. Hassan, S. Islam, N.S. Nafi, S. Bevinakoppa, Machine learning technologies for secure vehicular communication in internet of vehicles: recent advances and applications, Secur. Commun. Netw. 2021 (2021).
[13] N. Burkart, M.F. Huber, A survey on the explainability of supervised machine learning, J. Artificial Intelligence Res. 70 (2021) 245–317.
[14] A. Boualouache, T. Engel, A survey on machine learning-based misbehavior detection systems for 5G and beyond vehicular networks, 2022, arXiv preprint arXiv:2201.10500.
[15] D. Tsolkas, H. Koumaras, A.-S. Charismiadis, A. Foteas, Artificial intelligence in 5G and beyond networks, in: Applied Edge AI, Auerbach Publications, 2022, pp. 73–103.
[16] S. Rai, A.K. Garg, Applications of machine learning techniques in next-generation optical WDM networks, J. Opt. (2022) 1–10.
[17] M.A. Ridwan, N.A.M. Radzi, F. Abdullah, Y. Jalil, Applications of machine learning in networking: a survey of current issues and future challenges, IEEE Access (2021).
[18] A. Musa, I. Awan, Functional and performance analysis of discrete event network simulation tools, Simul. Model. Pract. Theory (2021) 102470.
[19] F. Wilhelmi, M. Carrascosa, C. Cano, A. Jonsson, V. Ram, B. Bellalta, Usage of network simulators in machine-learning-assisted 5g/6g networks, IEEE Wirel. Commun. 28 (1) (2021) 160–166.
[20] L. Campanile, M. Gribaudo, M. Iacono, F. Marulli, M. Mastroianni, Computer network simulation with ns-3: A systematic literature review, Electronics 9 (2) (2020) 272.
[21] P. Gawłowicz, A. Zubow, Ns3-gym: Extending openai gym for networking research, 2018, arXiv:1810.03943.
[22] e.a. Martin Abadi, Tensorflow: A system for large-scale machine learning, in: USENIX Symposium on Operating Systems Design and Implementation, 2016.
[23] S. Khastoo, T. Brecht, A. Abedi, Neura: Using neural networks to improve wifi rate adaptation, in: 23rd International ACM Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems, 2020, http://dx.doi.org/10.1145/3416010.3423217.
[24] W. Luo, X. Ma, L. Yang, A deep reinforcement learning-based handover scheme for vehicular networks, IEEE Access 8 (2020) http://dx.doi.org/10.1109/ACCESS.2020.3037162.
[25] S. Chen, Q. Zeng, J. Wu, A reinforcement learning-based admission control algorithm for multi-service hetnets, IEEE Access (2021) http://dx.doi.org/10.1109/ACCESS.2021.3063833.
[26] M.A. Alsheikh, S. Lin, D. Niyato, H.-P. Tan, Machine learning in wireless sensor networks: Algorithms, strategies, and applications, IEEE Commun. Surv. Tutor. 16 (4) (2014) 1996–2018.
[27] M. Alowish, Y. Shiraishi, M. Mohri, M. Morii, Three layered architecture for driver behavior analysis and personalized assistance with alert message dissemination in 5G envisioned Fog-IoCV, Future Internet 14 (1) (2022) 12.
[28] S.B. Prathiba, G. Raja, K. Dev, N. Kumar, M. Guizani, A hybrid deep reinforcement learning for autonomous vehicles smart-platooning, IEEE Trans. Veh. Technol. 70 (12) (2021) 13340–13350.
[29] J. Zang, M. Shikh-Bahaei, An adaptive full-duplex deep reinforcement learning-based design for 5G-V2X mode 4 VANETs, in: 2021 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2021, pp. 1–6.
[30] S.M. Hussain, K.M. Yusof, Dynamic Q-learning and fuzzy CNN based vertical handover decision for integration of DSRC, mmwave 5G and LTE in internet of vehicles (IoV)., J. Commun. 16 (5) (2021) 155–166.
[31] S.M. Hussain, K.M. Yusof, S.A. Hussain, Artificial intelligence-based network selection and optimized routing in internet of vehicles, Transp. Telecommun. J. 22 (4) (2021) 392–406.
[32] B. Sliwa, C. Schüler, M. Patchou, C. Wietfeld, PARRoT: Predictive ad-hoc routing fueled by reinforcement learning and trajectory knowledge, in: 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), IEEE, 2021, pp. 1–7.
[33] J. Aznar-Poveda, A.-J. Garcia-Sanchez, E. Egea-Lopez, J. García-Haro, Simultaneous data rate and transmission power adaptation in V2V communications: A deep reinforcement learning approach, IEEE Access 9 (2021) 122067–122081.
[34] G. Peserico, T. Fedullo, A. Morato, F. Tramarin, L. Rovati, S. Vitturi, Snr-based reinforcement learning rate adaptation for time critical Wi-Fi networks: Assessment through a calibrated simulator, in: 2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), IEEE, 2021, pp. 1–6.
[35] D. Wang, M. Wang, T. Zhang, S. Yang, C. Xu, An adaptive CMT-SCTP scheme: A reinforcement learning approach, J. Netw. Netw. Appl. 1 (4) (2022) 170–178.
[36] G.-P. Antonio, C. Maria-Dolores, AIM5LA: A latency-aware deep reinforcement learning-based autonomous intersection management system for 5G communication networks, Sensors 22 (6) (2022) 2217.
[37] T.-V.T. Duong, V.M. Ngo, et al., Reinforcement learning for QoS-guaranteed intelligent routing in wireless mesh networks with heavy traffic load, ICT Express (2022).
[38] C. Wang, H. Yan, F. Li, Self-selection protocol algorithm for wireless networks based on DDQN, in: Journal of Physics: Conference Series, 1871, IOP Publishing, 2021, 012134.
[39] R. Queiros, E. Almeida, H. Fontes, J. Ruela, R. Campos, Wi-Fi rate adaptation using a simple deep reinforcement learning approach, 2022, arXiv preprint arXiv:2202.03997.
[40] S. Cho, Reinforcement learning for rate adaptation in CSMA/CA wireless networks, in: Advances in Computer Science and Ubiquitous Computing, Springer, 2021, pp. 175–181.
[41] M. Drago, T. Zugno, F. Mason, M. Giordani, M. Boban, M. Zorzi, Artificial intelligence in vehicular wireless networks: A case study using ns-3, 2022, arXiv preprint arXiv:2203.05449.
[42] C. Schüler, M. Patchou, B. Sliwa, C. Wietfeld, Robust machine learning-enabled routing for highly mobile vehicular networks with PARRoT in NS-3, in: Proceedings of the Workshop on ns-3, 2021, pp. 88–94.
[43] S. Moon, S. Ahn, K. Son, J. Park, Y. Yi, Neuro-DCF: Design of wireless MAC via multi-agent reinforcement learning approach, in: Proceedings of the Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing, 2021, pp. 141–150.
[44] S. Mollahasani, M. Erol-Kantarci, M. Hirab, H. Dehghan, R. Wilson, Actor-critic learning based QoS-Aware scheduler for reconfigurable wireless networks, IEEE Trans. Netw. Sci. Eng. (2021).
[45] W. Wydmański, S. Szott, Contention window optimization in IEEE 802.11 ax networks with deep reinforcement learning, in: 2021 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2021, pp. 1–6.
[46] G. Alsuhli, H.A. Ismail, K. Alansary, M. Rumman, M. Mohamed, K.G. Seddik, Deep reinforcement learning-based CIO and energy control for LTE mobility load balancing, in: 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2021, pp. 1–6.
[47] G. Alsuhli, K. Banawan, K. Seddik, A. Elezabi, Optimized power and cell individual offset for cellular load balancing via reinforcement learning, in: 2021 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2021, pp. 1–7.
[48] P.E. Iturria-Rivera, M. Erol-Kantarci, Qos-aware load balancing in wireless networks using clipped double Q-learning, in: 2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS), IEEE, 2021, pp. 10–16.
[49] M.S.A. Muthanna, A. Muthanna, A. Rafiq, M. Hammoudeh, R. Alkanhel, S. Lynch, A.A. Abd El-Latif, Deep reinforcement learning based transmission policy enforcement and multi-hop routing in QoS aware LoRa IoT networks, Comput. Commun. 183 (2022) 33–50.
[50] D.A. AlWahab, G. Gombos, S. Laki, On a deep Q-network-based approach for active queue management, in: 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), IEEE, 2021, pp. 371–376.

[51] M. Elsayed, R. Joda, H. Abou-Zeid, R. Atawia, A.B. Sediq, G. Boudreau, M. Erol-Kantarci, Reinforcement learning based energy-efficient component carrier activation-deactivation in 5G, in: 2021 IEEE Global Communications Conference (GLOBECOM), IEEE, 2021, pp. 1–6.

[52] R. Ali, I. Ashraf, A.K. Bashir, Y.B. Zikria, Reinforcement-learning-enabled massive internet of things for 6G wireless communications, IEEE Commun. Stand. Mag. 5 (2) (2021) 126–131.

[53] F. Wu, W. Yang, J. Lu, F. Lyu, J. Ren, Y. Zhang, RLSS: A reinforcement learning scheme for HD map data source selection in vehicular NDN, IEEE Internet Things J. (2021).

[54] M. Aboelwafa, G. Alsuhli, K. Banawan, K.G. Seddik, Self-optimization of cellular networks using deep reinforcement learning with hybrid action space, in: 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2022, pp. 223–229.

[55] M.M. Medvei, G.-A. Dima, N. Ţăpuş, Approaching traffic congestion with double deep Q-networks, in: 2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet), IEEE, 2021, pp. 1–6.

[56] A.K.C.S. Boni, H. Hassan, K. Drira, Task offloading in autonomous IoT systems using deep reinforcement learning and ns3-gym, in: 11th International Conference on the Internet of Things (IoT 2021), 2021.

[57] R. Pradeep, A. Babu, K. Midhula, A.R.K. Parthiban, Temporal difference learning model for TCP end-to-end congestion control in heterogeneous wireless networks, in: 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), IEEE, 2021, pp. 1–5.

[58] L. Bonati, S. D'Oro, M. Polese, S. Basagni, T. Melodia, Intelligence and learning in O-RAN for data-driven NextG cellular networks, IEEE Commun. Mag. 59 (10) (2021) 21–27.

[59] L. Bonati, P. Johari, M. Polese, S. D'Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder, et al., Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation, in: 2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), IEEE, 2021, pp. 105–113.

[60] L. Bonati, M. Polese, S. D'Oro, S. Basagni, T. Melodia, Intelligent closed-loop RAN control with xapps in OpenRAN gym, 2022, arXiv preprint arXiv:2208.14877.

[61] M. Polese, L. Bonati, S. D'Oro, S. Basagni, T. Melodia, ColO-RAN: Developing machine learning-based xapps for open RAN closed-loop control on programmable experimental platforms, IEEE Trans. Mob. Comput. (2022).

[62] B. Casasole, L. Bonati, S. D'Oro, S. Basagni, A. Capone, T. Melodia, Qcell: Self-optimization of softwarized 5G networks through deep Q-learning, in: 2021 IEEE Global Communications Conference (GLOBECOM), IEEE, 2021, pp. 01–06.

[63] J.M. Shea, T.F. Wong, A deep Q-learning dynamic spectrum sharing experiment, in: ICC 2021-IEEE International Conference on Communications, IEEE, 2021, pp. 1–6.

[64] C. Bowyer, D. Greene, T. Ward, M. Menendez, J. Shea, T. Wong, Reinforcement learning for mixed cooperative/competitive dynamic spectrum access, in: 2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), IEEE, 2019, pp. 1–6.

[65] C. Song, W. Xu, T. Wu, S. Yu, P. Zeng, N. Zhang, QoE-driven edge caching in vehicle networks based on deep reinforcement learning, IEEE Trans. Veh. Technol. 70 (6) (2021) 5286–5295.

[66] R. Hu, X. Wang, Y. Su, B. Yang, An efficient deep reinforcement learning based distributed channel multiplexing framework for V2X communication networks, in: 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE), IEEE, 2021, pp. 154–160.

[67] J. Mei, X. Wang, K. Zheng, G. Boudreau, A.B. Sediq, H. Abou-Zeid, Intelligent radio access network slicing for service provisioning in 6G: A hierarchical deep reinforcement learning approach, IEEE Trans. Commun. 69 (9) (2021) 6063–6078.

[68] M. Zambianco, G. Verticale, A reinforcement learning agent for mixed-numerology interference-aware slice spectrum allocation with non-deterministic and deterministic traffic, Comput. Commun. (2022).

[69] E. Kim, B.C. Jung, C.Y. Park, H. Lee, Joint optimization of energy efficiency and user outage using multi-agent reinforcement learning in ultra-dense small cell networks, Electronics 11 (4) (2022) 599.

[70] A. Paul, S.P. Maity, Reinforcement learning based Q-routing: Performance evaluation on cognitive radio network topologies, Wirel. Pers. Commun. (2022) 1–17.

[71] Z. Guo, Z. Chen, P. Liu, J. Luo, X. Yang, X. Sun, Multi-agent reinforcement learning based distributed channel access for next generation wireless networks, IEEE J. Sel. Areas Commun. (2022).

[72] J. Jia, R. Yu, Z. Du, J. Chen, Q. Wang, X. Wang, Distributed localization for IoT with multi-agent reinforcement learning, Neural Comput. Appl. (2022) 1–14.

[73] J. Tan, W. Guan, Resource allocation of fog radio access network based on deep reinforcement learning, Eng. Rep. (2022) e12497.

[74] A. Bitaillou, B. Parrein, G. Andrieux, K. Couty, Latency filtering for Q-routing on wireless networks, in: 2021 International Wireless Communications and Mobile Computing (IWCMC), IEEE, 2021, pp. 1314–1319.

[75] T.G. Nguyen, T.V. Phan, D.T. Hoang, H.H. Nguyen, D.T. Le, Deepplace: Deep reinforcement learning for adaptive flow rule placement in software-defined IoT networks, Comput. Commun. 181 (2022) 156–163.

[76] F. Majidi, M.R. Khayyambashi, B. Barekatain, Hfdrl: An intelligent dynamic cooperate cashing method based on hierarchical federated deep reinforcement learning in edge-enabled iot, IEEE Internet Things J. 9 (2) (2021) 1402–1413.

[77] K.-C. Chiu, C.-C. Liu, L.-D. Chou, Reinforcement learning-based service-oriented dynamic multipath routing in SDN, Wirel. Commun. Mob. Comput. 2022 (2022).

[78] J.W. Lim, Y.-B. Ko, Q-learning based stepwise routing protocol for multi-UAV networks, in: 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), IEEE, 2021, pp. 307–309.

[79] X. Zhang, M. Lei, C. Wang, M. Zhao, A transmission and backoff method based on deep reinforcement learning for statistical priority-based multiple access network, in: 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), IEEE, 2021, pp. 1–5.

[80] A.M. Pasikhani, J.A. Clark, P. Gope, Reinforcement-learning-based ids for 6low-pan, in: 2021 IEEE 20th International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom), IEEE, 2021, pp. 1049–1060.

[81] J. Nunez-Martinez, M. Portolès-Comeras, J. Nin-Guerrero, P. Dini, N. Baldo, M. Requena-Esteso, J. Mangues-Bafalluy, Validation of the IEEE 802.11 MAC model in the ns3 simulator using the extreme testbed, 2010.

[82] S.S.M. Malekzadeh, A. Ghani, J. Desa, Validating reliability of OMNeT++ in wireless networks DoS attacks: Simulation vs. testbed, 13 (2011) 13–21.

[83] G. Kremer, D. Carvin, P. Berthou, P. Owezarski, Configuration schemes and assessment of NS3 models using a wireless testbed, 2013.

[84] NS-3, NS-3 source code, 2022, https://github.com/nsnam/ns-3-dev-git, [Online; accessed March-2022].

[85] OMNeT++, OMNeT++ models, 2022, https://omnetpp.org, [Online; accessed March-2022].

[86] Y. Sun, M. Peng, S. Mao, Deep reinforcement learning-based mode selection and resource management for green fog radio access networks, IEEE Internet Things J. 6 (2) (2018) 1960–1971.

[87] K. Selvaraju, A. Kumar Saha, Ramdoot Pydipaty, Johnu George, Improving the performance of deep learning for wireless localization, 2020.

[88] V. Nguyen, Bayesian optimization for accelerating hyper-parameter tuning, in: IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), 2019.

[89] A.K. Snehal Chitnavis, Cross layer routing in cognitive radio networks using deep reinforcement learning, 2019.

Serap Ergün is Assistant Professor at the Isparta University of Applied Sciences since 2021. She was a post-doc researcher in 2022 and 2023 at the University Clermont Auvergne. Her research interests include cooperative game theory, network routing algorithms, network simulations, intelligent agents, agent development platforms, supply chain management, cloud computing.

Ibrahim Sammour is currently pursuing a Ph.D. degree in computer science at University Clermont Auvergne, Clermont-Ferrand, France. His research interests include deep reinforcement learning, vehicular networks, IEEE 802.11 networks, and next-generation networks.

Gerard Chalhoub is Associate Professor at the University Clermont Auvergne and a member of the LIMOS (Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes). His research interests lie in the field of Internet of Things, agriculture, decision making, intelligent transportation systems, radio access networks, telecommunication standards, vehicular ad hoc networks, Long Term Evolution, cellular radio, design engineering, ecology, mobile robots, mobility management, quality of service, road safety, scheduling, telecommunication computing, telecommunication congestion control, telecommunication network reliability, telecommunication networks, telecommunication traffic, ubiquitous computing, wireless channels, wireless sensor networks.