Journal of Network and Computer Applications 216 (2023) 103669

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Review

A comprehensive survey on reinforcement-learning-based computation offloading techniques in Edge Computing Systems

Diego Hortelano [a,b,*], Ignacio de Miguel [a], Ramón J. Durán Barroso [a], Juan Carlos Aguado [a], Noemí Merayo [a], Lidia Ruiz [a], Adrian Asensio [c], Xavi Masip-Bruin [c], Patricia Fernández [a], Rubén M. Lorenzo [a], Evaristo J. Abril [a]

[a] Universidad de Valladolid, Valladolid, Spain
[b] Universidad Rey Juan Carlos, Móstoles, Spain
[c] Universidad Politècnica de Catalunya, Barcelona, Spain

ARTICLE INFO

Keywords:
Computation offloading
Edge computing
MEC
Multi-Access Edge Computing
Reinforcement Learning
Deep Reinforcement Learning

ABSTRACT

In recent years, the number of embedded computing devices connected to the Internet has exponentially increased. At the same time, new applications are becoming more complex and computationally demanding, which can be a problem for devices, especially when they are battery powered. In this context, the concepts of computation offloading and edge computing, which allow applications to be fully or partially offloaded and executed on servers close to the devices in the network, have arisen and received increasing attention. Then, the design of algorithms to make the decision of which applications or tasks should be offloaded, and where to execute them, is crucial. One of the options that has been gaining momentum lately is the use of Reinforcement Learning (RL) and, in particular, Deep Reinforcement Learning (DRL), which enables learning optimal or near-optimal offloading policies adapted to each particular scenario. Although the use of RL techniques to solve the computation offloading problem in edge systems has been covered by some surveys, it has been done in a limited way. For example, some surveys have analysed the use of RL to solve various networking problems, with computation offloading being one of them, but not the primary focus. Other surveys, on the other hand, have reviewed techniques to solve the computation offloading problem, being RL just one of the approaches considered. To the best of our knowledge, this is the first survey that specifically focuses on the use of RL and DRL techniques for computation offloading in edge computing system. We present a comprehensive and detailed survey, where we analyse and classify the research papers in terms of use cases, network and edge computing architectures, objectives, RL algorithms, decision-making approaches, and time-varying characteristics considered in the analysed scenarios. In particular, we include a series of tables to help researchers identify relevant papers based on specific features, and analyse which scenarios and techniques are most frequently considered in the literature. Finally, this survey identifies a number of research challenges, future directions and areas for further study.

1. Introduction

In recent years, the trend of embedding computing capabilities into everyday objects, known as pervasive computing, is significantly increasing. Thus, the number of devices with limited computing resources such as Internet of Things (IoT) devices, mobile devices, vehicles or Unmanned Aerial Vehicles (UAVs), among others, which require executing applications to process data is growing. Moreover, these applications are becoming more and more computationally demanding, complicating their local execution (Yan et al., 2020).

In order to solve the processing limitation of these devices, the concept of computation offloading emerged. This allows applications to be offloaded to data centres with higher computational capacities in order to be executed in them, greatly reducing execution time. This concept was originally linked to Cloud Computing, which is defined by the National Institute of Standards and Technology (NIST) as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (Mell and Grance, 2011). These resources are provided by remote data centre resources over the Internet (Vahid Dastjerdi et al., 2016). However, the distance at which these data centres are located increases the delay in communications, causing the offloading time of applications to

* Corresponding author at: Universidad Rey Juan Carlos, Móstoles, Spain.
E-mail addresses: diego.hortelano@tel.uva.es, diego.hortelano@urjc.es (D. Hortelano).

https://doi.org/10.1016/j.jnca.2023.103669
Received 21 October 2022; Received in revised form 27 February 2023; Accepted 4 May 2023
Available online 8 May 2023

become longer, which may not be possible or reasonable for real-time applications or those with strong latency requirements. Recently, the emergence of concepts such as Mobile Cloud Computing (MCC), Edge Computing, Fog Computing and Multi-Access Edge Computing (MEC) has provided a way to deal with the latency problems associated with Cloud Computing. Although the objective of these paradigms is similar: to enable the offloading of applications for their execution in proximate computational resources rather than in remote data centres, in order to reduce application latency and network traffic, they are not entirely equivalent, and there are some differences, as mentioned in Yousefpour et al. (2019) and Ferrer et al. (2019).

The concept of MCC first appeared in 2009 (Satyanarayanan et al., 2009), being the precursor of related paradigms such as Fog and Edge Computing. MCC comes from the combination of Cloud Computing with Mobile Computing (which refers to computing performed on mobile or portable devices) (Yousefpour et al., 2019), and is defined in Dinh et al. (2013) as an infrastructure where data storage and processing occurs in the cloud, outside of mobile devices, providing mobile users with access to applications and services over the Internet. The main motivation for this paradigm was to increase the computing resources accessible to mobile devices. There is a variation of MCC, known as Mobile Ad hoc Cloud Computing (MACC), which also makes use of the resources available on other mobile devices for the execution of offloaded tasks (Ferrer et al., 2019).

Although the current concept of Edge Computing is largely based on MCC, its origins can be found in the Content Distributed Networks (CDN) created in the late 1990s to serve web and video content from servers placed close to users (Dilley et al., 2002), which later evolved into edge servers running applications, being the first commercial Edge Computing service (Davis et al., 2004; Nygren et al., 2010). Thus, Edge Computing enables the storage and processing of data generated by devices (mainly IoT devices) in small data centres located close to these devices, at the edge of the network (Yousefpour et al., 2019; Shi et al., 2016). Note that the edge of the network is not located at the devices, but as close as one hop away from them (or, at most, one hop away from the local network) (Yousefpour et al., 2019). This approach from resources to data sources has a clear objective: to reduce the distance that data travels, thus reducing processing latency and network traffic.

Fog Computing was first defined in 2012 by Bonomi et al. (2012), choosing the term fog because fog is a cloud closer to the ground. OpenFog Consortium (OpenFog Consortium, 2022a) defines Fog as a horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum. Furthermore, with this definition, the OpenFog Consortium emphasises the difference between fog and edge, restricting the last one to computing at the edge of the network. Thus, Fog Computing can be considered as a bridge between the cloud and end devices that allows computing, storage, networking, decision-making and data management to take place not only in the cloud, but also along the path between devices and the cloud, prioritising proximity to devices (Yousefpour et al., 2019). Additionally, researchers have proposed multiple definitions for this term (Bonomi et al., 2012; Alhaddadin et al., 2014; Vaquero and Rodero-Merino, 2014). The Fog Computing platform permits to reduce the load of traditional Cloud Computing data centres, supporting geographically distributed, latency-sensitive and Quality of Service (QoS)-aware applications (Mukherjee et al., 2018) and providing better and faster applications (Naha et al., 2018).

Another concept related to the Edge Computing paradigm is MEC, which is being developed by an Industry Specification Group (ISG) of the European Telecommunications Standards Institute (ETSI) (Hu et al., 2015), with the aim of creating a standard that integrates Edge Computing (providing Cloud Computing functionality) into the mobile network architecture. Behind these standardisation efforts are large mobile operators such as DOCOMO, Vodafone or TELECOM Italia, and manufacturers such as IBM, Nokia, Huawei or Intel (Mach and Becvar, 2017). Due to its focus on mobile networks, MEC initially stood for Mobile Edge Computing. However, in September 2017, ETSI increased the scope of MEC to also consider the edge of non-mobile networks. Thus, it changed the term "Mobile" to "Multi-Access". In this way, the original acronym, MEC, was retained, but renamed to Multi-access Edge Computing (Anon, 2017). Similar to how MCC can be seen as the extension of Mobile Computing through Cloud Computing, MEC can be seen as the extension of Mobile Computing through Edge Computing (Yousefpour et al., 2019). ETSI defined MEC as a platform that provides Cloud Computing capabilities in the Radio Access Network (RAN) in 4G and 5G, close to the users (Giust et al., 2018). MEC proposes the location of compute and storage resources at the base stations of cellular networks (Beck et al., 2014), allowing RAN operators to add Edge Computing functionality to existing base station. Moreover, MEC supports two or three layers of hierarchical application development allowing the use of small-scale and cloud data centres (Klas, 2015).

As discussed above, MCC, Edge Computing, Fog Computing and MEC are different concepts, although they all have a similar objective: to reduce the execution latency of offloaded tasks. Therefore, although the OpenFog Consortium and ETSI have clarified the differences between the different terms, it is common for authors to interchange these terms, as mentioned in Ferrer et al. (2019) and as we have confirmed personally during the conduct of this survey. Furthermore, technology providers have adopted the term Edge Computing in the discussion on what term to use (Ferrer et al., 2019). Therefore, like them, in this survey we will use the term Edge Computing to refer to all those architectures and environments proposed by researchers that provide execution resources and are placed between the source of the data and the cloud infrastructure. However, as this is a literature review, we will use whatever term the author refers to, including MCC, Edge Computing, Fog Computing or MEC.

In this context, algorithms that decide where to execute each of the applications of the devices (either in the cloud, in edge computing nodes or in the device itself) should be designed. Thus, a number of techniques have been proposed and used to solve the computation offloading problem in these environments. These methods include algorithms based on nonlinear programming (Chen and Hao, 2018), mixed integer programming (Alameddine et al., 2019), the interior point method (Vu et al., 2018), the branch and bound method (Vu et al., 2018), greedy heuristics (Sun and Nakhai, 2020), heuristic algorithms (Yang et al., 2018) and machine learning techniques (Shakarami et al., 2020b).

Within machine learning techniques, the use of Reinforcement Learning (RL) has recently received a great deal of attention from the research community. RL is a branch of machine learning that allows learning how to act based on the interaction with the environment, and taking into account that the actions performed not only have immediate outcomes and impact on the environment but also for the future. For that reason, RL is a very suitable technique for learning control policies in complex and dynamic environments like those of edge computing systems. Edge computing systems usually involve a large (and variable) number of end devices and network nodes, which communicate over wireless and wired communication links. Applications or tasks with different processing and latency requirements are dynamically generated, so the total load faced by the network fluctuates over time. In addition, the network must efficiently cope with the mobility of devices (and even servers in some cases). Therefore, the complexity and dynamism of these scenarios make managing resources, and deciding which tasks to offloaded and where, not straightforward. Moreover, in these cases, real-time decisions need to be made, and incorrect decisions can lead to a degradation of the efficiency of the system. Although in simple, low-dynamic scenarios a solution can be found using traditional algorithms, they usually do not allow efficient resolution in complex environments with high dynamism and variability. In contrast, RL algorithms have the potential to adapt to these dynamic scenarios thanks to the interaction with the environment, and improve their policies based on past experience and the cumulative

rewards received from past actions. To do this, RL agents must explore, i.e., try new actions in order to make a better selection of actions in the future, but also exploit what they have already learnt so far, so there is a trade-off between these exploration and exploitation activities. Thus, RL algorithms are complex in themselves. They require extensive training through interaction with an environment and, since many wrong decisions will be made at the beginning of such training, a simulated offline environment is usually employed, at least in the initial phases of the method. Moreover, an RL algorithm must be able to generalise, i.e., to provide appropriate decisions even when facing situations different from those it has previously encountered, and for that reason, RL is usually combined with deep learning techniques. In summary, since RL algorithms can learn optimal or near-optimal policies to make computing offloading decisions, a multitude of RL and Deep Reinforcement Learning (DRL)-based solutions have appeared in recent years.

The use of RL for computation offloading have been addressed by previous surveys in a limited manner. Some have focused on the use of RL techniques to solve various networking problems and include only a few papers on computation offloading, while others have taken the opposite approach and focused on techniques specifically proposed for computation offloading, with only limited coverage of RL techniques. The most relevant survey related to the problem addressed in this paper concentrates on the use of machine learning techniques (including RL, supervised learning, and unsupervised learning) in computation offloading problems, but it only includes articles published up until the beginning of 2020. This situation is discussed in detail in Section 4. Despite the growing momentum around the use of RL techniques to address computation offloading in edge systems (as evidenced by the number of articles included in this survey), to the best of our knowledge, there is currently no survey specifically focused on this issue. Our aim with this survey is to assist researchers by providing a comprehensive and detailed analysis of current solutions, complemented by a set of tables that facilitate the identification of the most relevant papers based on the scenarios of interest for each researcher. In addition, we provide future research directions and areas for further study.

1.1. Main contributions

In this paper, we present a comprehensive survey of the application of RL techniques to the problem of computation offloading in edge computing systems. There are several previous surveys on RL applied to computational offloading; however, they only review papers up to the beginning of 2020, as we will discuss in Section 4. This paper aims to fill that gap by reviewing most recent papers. The survey reviews and classifies the solutions in terms of different use cases, network and edge computing architectures, RL algorithms used, objectives and performance metrics, decision-making approaches (either centralised or distributed), number of applications and partitioning considerations, time-varying conditions in the scenarios, and type of evaluation of the proposals. Thus, the main contributions of this paper are:

- To review current research papers proposing RL-based solutions for computation offloading in edge computing systems.
- To explore in depth RL-based solutions for computation offloading, as well as the scenarios considered by the authors.
- To provide a systematic review of current solutions, classifying them according to different aspects, in order to organise a large number of papers into a knowledge structure. In particular, the paper provides a set of tables (Tables 3 to 8), which summarise the main features of the reviewed papers.
- To determine which network architectures, RL algorithms, decision-making approaches and metrics are more frequently employed in the literature.
- To expose the research challenges and problems to be solved in future research, as well as the areas that need more basic research.

Table 1
List of acronyms.

Acronym	Description
A2C	Synchronous Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-Critic
AC	Actor-Critic
AP	Access Point
BS	Base Station
DAG	Directed Acyclic Graph
DDQN	Double Deep Q Network
DDPG	Deep Deterministic Policy Gradient
DNN	Deep Neural Network
DQN	Deep Q Network
DRL	Deep Reinforcement Learning
ETSI	European Telecommunications Standards Institute
IoT	Internet of Things
IoV	Internet of Vehicles
KNN	K-Nearest Neighbour
MAB	Multi Armed Bandit
MACC	Mobile Ad hoc Cloud Computing
MC	Monte Carlo
MCC	Mobile Cloud Computing
MDP	Markov Decision Process
MEC	Multi-Access Edge Computing
ML	Machine Learning
NOMA	Non-Orthogonal Multiple Access
PPO	Proximal Policy Optimisation
PG	Policy Gradient
QL	Q-Learning
QoS	Quality of Service
RL	Reinforcement Learning
RQ	Research Question
RSU	Roadside Unit
SAC	Soft Actor Critic
SARSA	State-Action-Reward-State-Action
SDN	Software-Defined Networking
TD	Temporal Difference
TD3	Twin Delayed DDPG
UAV	Unmanned Aerial Vehicle
VR	Virtual Reality

Moreover, in the next section (Section 2) we describe a set of research questions addressed in this work. The answer to those research questions is another contribution of this survey.

1.2. Organisation of the survey

The rest of this article is organised as follows. First of all, Section 2 describes the research questions that we address in this survey, and the methodology that we have used in the search for articles to review. Then, Section 3 provides a summary of key concepts necessary to understand the rest of the paper and the taxonomy employed to classify the articles, including a short introduction to topics like reinforcement learning, networking environments typically considered in edge computing systems, and common objectives when approaching offloading tasks. Next, Section 4 presents related surveys. To the best of our knowledge, the latest published survey focusing on RL solutions for computation offloading in edge computing systems only includes articles published up to early 2020. Therefore, Section 5 reviews and classifies recent research in the area, including articles published between 2020 and 2021. Finally, Section 6 provides an analysis and discussion of the reviewed articles, including research challenges and future directions, and Section 7 presents the conclusions of this survey. The list of the most common acronyms used in this article is presented in Table 1.

2. Research questions and methodology

This section first presents the research questions addressed in this work. Then, it describes the review protocol followed for the inclusion of articles in this survey.

2.1. Research questions

This survey explores current articles that address the problem of computation offloading in edge computing systems, identifying the main aspects that the authors are taking into account when modelling the environment, as well as the RL-based solutions being used. In order to carry out a comprehensive study of these articles and to present open issues, the following Research Questions (RQs) have been considered:

- **RQ1 (use cases)**: In which use cases, related to computation offloading and edge computing, are RL approaches used?
- **RQ2 (network and edge computing architectures)**: Which network and edge computing architecture is considered? Are there multiple end devices and edge servers? Is a cloud layer also considered?
- **RQ3 (algorithms)**: Which type of RL-based algorithms are usually employed for computation offloading?
- **RQ4 (objectives)**: What are the metrics typically optimised when solving computation offloading by means of RL techniques?
- **RQ5 (centralised vs. distributed)**: Are centralised or distributed approaches generally used for decision-making?
- **RQ6 (number of applications and partitioning)**: Do authors consider that end devices which offload their applications to an edge computing system run a single application or multiple applications? Are these applications to be fully executed on the same platform or can they be split? If they are divided into smaller tasks, is the data dependency among the tasks of the same application taken into account?
- **RQ7 (time-varying aspects)**: What characteristics of the environment are considered to be time-varying in the modelled scenarios?
- **RQ8 (evaluation)**: How did the authors evaluate and verify their proposals? Did they conduct experiments using simulations or were they conducted in testbeds or real environments?
- **RQ9 (future directions)**: What are the future directions of research in this field and what areas remain to be addressed?

2.2. Review protocol

As previously mentioned, we have explored articles in which the authors apply RL to solve the computation offloading problem in edge computing systems. First, we have searched for related surveys in the topic (which are discussed in Section 4), and then, we have conducted a systematic search to retrieve recent articles in this area (which are discussed in Section 5). These papers have been obtained from the Scopus database (Elsevier, 2022b), in which the following article selection criteria were followed:

- Articles published in 2020 and 2021.
- Articles which include the following terms in their title, abstract or keywords: (“computation offloading”) AND (“reinforcement learning”) AND ((“mec”) OR (“edge”)).

The search was conducted in January 1st, 2022, resulting in a total of 81 results for the year 2020 and 74 for the year 2021. For the year 2020, there were a total of 7 articles not included because they were not really related to the scope of this survey, while for the year 2021 there were 8 excluded articles. Thus, the total number of included articles was 140.

3. Background and categories for the classification of the literature

This section presents the summarised background necessary to understand the rest of the article, as well as the taxonomy and different categories used to classify the literature. It has been is divided into several subsections, which are related to the main research questions previously enumerated.

3.1. Use cases (RQ1)

Computation offloading in edge computing systems has been analysed in different use cases or scenarios, like IoT networks, vehicular networks, UAVs, virtual reality or robotics, as well as in generic studies, due to the different objectives and characteristics of the diverse use cases, as detailed in Section 5. Precisely, the type of case study will be the first element that we will use to classify the different works on the area. Thus, a different table has been created for each case study (Tables 3 to 8), which summarise the main features of the papers associated with each of those case studies.

3.2. Network and edge computing architecture (RQ2)

The generic network architecture considered in edge computing systems usually has at least two layers: the end-device layer and the edge layer. Nevertheless, a third layer, the cloud layer, is also sometimes considered. Moreover, within each of those layers, different studies make different assumptions:

- *End-device layer*: This layer is usually composed by devices with low computational capacity and limited power, like IoT devices or mobile devices. These devices are responsible for collecting data, processing it and performing actions. However, data size and processing requirements are continuously increasing. Although some papers consider just one device, most of the studies assume scenarios with multiple devices within this layer.
- *Edge layer*: The edge layer consists of a set of servers or data centres with high computing capacities where end devices can offload computation tasks, but they are located close to the end devices in order to reduce latency. This layer not only includes the servers or processing units, but also the Base Stations (BSs) or Access Points (APs) that allow end devices to use the computing resources. In fact, these BSs or APs can either contain the computing resources themselves or only give access to a server(s) providing the computing resources. Moreover, in some cases, a hierarchical set of computing resources is also considered: a first level of edge computing resources located at the BSs or APs, and a second level in a nearby data centre.
- *Cloud layer*: This layer consists of data centres characterised by much higher computational and storage capabilities. However, these powerful resources are located at a greater distance from the end user, which implies higher latencies. Therefore, this layer is an ideal choice for running computationally intensive tasks that do not have strict requirements for low latency. Nevertheless, not all the studies on edge computing systems consider this layer.

An example of a three-layer edge network architecture is shown in Fig. 1, which includes two levels of computing resources in the edge layer.

Tables 3 to 8 in Section 5 show, in their *Network Architecture* column, the network structure assumed in the reviewed articles, stating whether its end-device layer is single-device or multi-device; whether the edge layer has one or several BSs and edge servers; and whether the cloud layer is present or not.

3.3. RL algorithms (RQ3)

In order to make computation offloading decisions in the network architectures previously described, different RL and mainly DRL techniques have been proposed. This subsection gives a brief overview of these techniques. This summary has been written using Sutton and Barto (2018) and Dong et al. (2020) as main sources, which can be consulted for a more in-depth explanation.

RL is a subset of machine learning whose algorithms are characterised by learning through interaction with an environment. In a

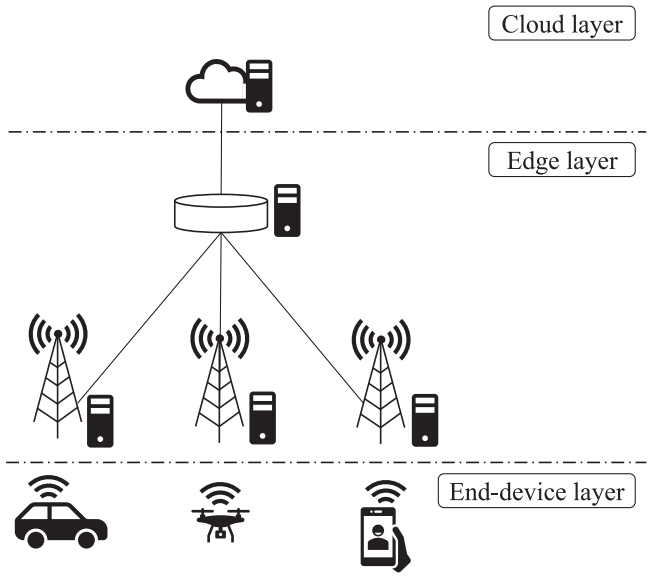

Fig. 1. Example of a three-layer edge network architecture.

nutshell, RL algorithms divide the world into an *environment* and an *agent*. The agent observes the *state* of the environment and interacts with it by executing specific actions which are decided following a *policy*, π. Then, the agent receives a certain *reward* from the environment, and the environment typically evolves to a different *state*. By means of these interactions, the agent can learn to make better decisions (i.e., to improve the policy) by trying to obtain better rewards from the environment, with the goal of maximising the cumulative reward, which is know as the *return*.

Value functions are typically employed when solving RL problems. The state-value function, $v_\pi(s)$, is the expected return when the agent starts from state s, and then acts following policy π. Similarly, the action-value function $q_\pi(s,a)$ is the expected return when the agent starts from state s, takes action a, and then follows policy π. RL aims to find the optimal policy, i.e., the policy which maximises the previously mentioned value functions.

In some cases, a *model* of the environment is available, which means that the agent can predict how the environment will respond to its actions (i.e., which immediate reward it will receive, and what are the possible next states and with which probabilities). Then, based on that information, *planning* techniques can be exploited in order to learn the optimal (or near-optimal) policy.

However, in many cases such a model is not known. In those cases, model-free RL techniques like Monte Carlo (MC) or Temporal Difference (TD) are used. The key idea is to start with an arbitrary policy, evaluate the value function, and then improve the policy based on the outcome of the value function evaluation (but leaving room for the exploration of other alternatives), and iterate through this process of evaluation and policy improvement.

MC and TD methods learn from experience, i.e., they learn the value function from samples of sequences of states, actions, and rewards from real or simulated interactions with the environment. However, MC techniques assume that the experience is divided into episodes, and that all episodes eventually terminate. Then, they estimate the value function by averaging the returns obtained in those complete episodes. While in many cases this is a valid approach, the communication networks where computation offloading takes place are operating continuously, so the episodic assumption is not met. Although one option is to artificially split the experience of continuous operation into episodes (for instance, after a certain number of steps), another possibility is the use of TD methods. They also learn from experience, but they are able to learn in each step (without waiting for a final event) thanks to the use of bootstrapping. This technique consists in estimating the value function of a state–action by taking into account the immediate reward and also the estimates of the value function of other states–actions. Two well known examples of TD techniques are Sarsa and Q-learning (QL). Sarsa is an on-policy method, which means that it evaluates and improves the very same policy that is used to make decisions. In contrast, Q-learning is an off-policy method. That means that it evaluates an improves a policy which is different from that used to make decisions. In particular, Q-learning enables learning an optimal policy (where no random actions are made for exploration purposes) while following a policy which eventually makes random actions for exploration purposes.

RL techniques, like Sarsa and Q-learning, have been classically implemented as tabular methods, where tables store the value function of each state or state–action. However, that approach does not scale well. For instance, in edge and networking environments, due to the high number of potential states, tabular approaches are not feasible. Rather than using tables, the key idea is to estimate value functions by means of function approximation techniques, i.e., by means of supervised learning. In this way, the value function is approximated by a parameterised function with depends on a number of weights. For instance, the value function can be approximated by a linear combination of features (which determine the state of the system), or by neural or Deep Neural Network (DNN). In particular, the combination of deep learning methods with Q-learning leads to DRL techniques like Deep Q Network (DQN), Double Deep Q Network (DDQN) or Dueling DQN.

The methods that we have discussed so far to learn the optimal policy were based on first estimating the value function. Another strategy that is also frequently used consists in directly learning a parameterised policy, so that it can select actions without consulting a value function. These are the Policy Gradient (PG) methods. When policy gradient methods are used, a value function may still be used to learn the policy parameters, but is not required for action selection. Examples of policy gradient methods are Actor-Critic (AC), Synchronous Advantage Actor-Critic (A2C), Asynchronous Advantage Actor-Critic (A3C) and Deep Deterministic Policy Gradient (DDPG).

Fig. 2 shows a taxonomy of the main RL algorithms. Interested readers may consult (Sutton and Barto, 2018; Dong et al., 2020; Zhang and Yu, 2020) for further details on these algorithms.

Tables 3 to 8 in Section 5 show, in their *Algorithm* column, the RL algorithms used by the authors of all the reviewed articles to solve the computation offloading problem in their proposed scenarios.

3.4. Objectives and performance metrics (RQ4)

Deciding on which node to execute or offload a computing application is an optimisation problem; thus, an objective to maximise or minimise should be defined to drive these decisions. Once the objective has been defined, the rewards of the RL algorithm should be set consistently with that objective, so that the maximisation of the cumulative reward (as described in Section 3.3) should, ideally, lead to getting the optimal value of the objective function or metric.

One of the most commonly used metrics is the *latency* or execution delay of applications, being the objective to minimise it. This delay is given by the execution time of the application and the transmission time if the application is offloaded. Closely related to latency minimisation is the objective of maximising *QoS*, although this can also include application prioritisation and application execution guarantees.

One of the problems of considering the *minimisation of latency or application delay* as the objective is that this goal could be obtained at the expense of the execution of other applications. For this reason, some authors choose to provide a negative reward in case the required latency is not fulfilled, while some other authors establish the objective of *maximising the computation rate* or the *number of completed applications*.

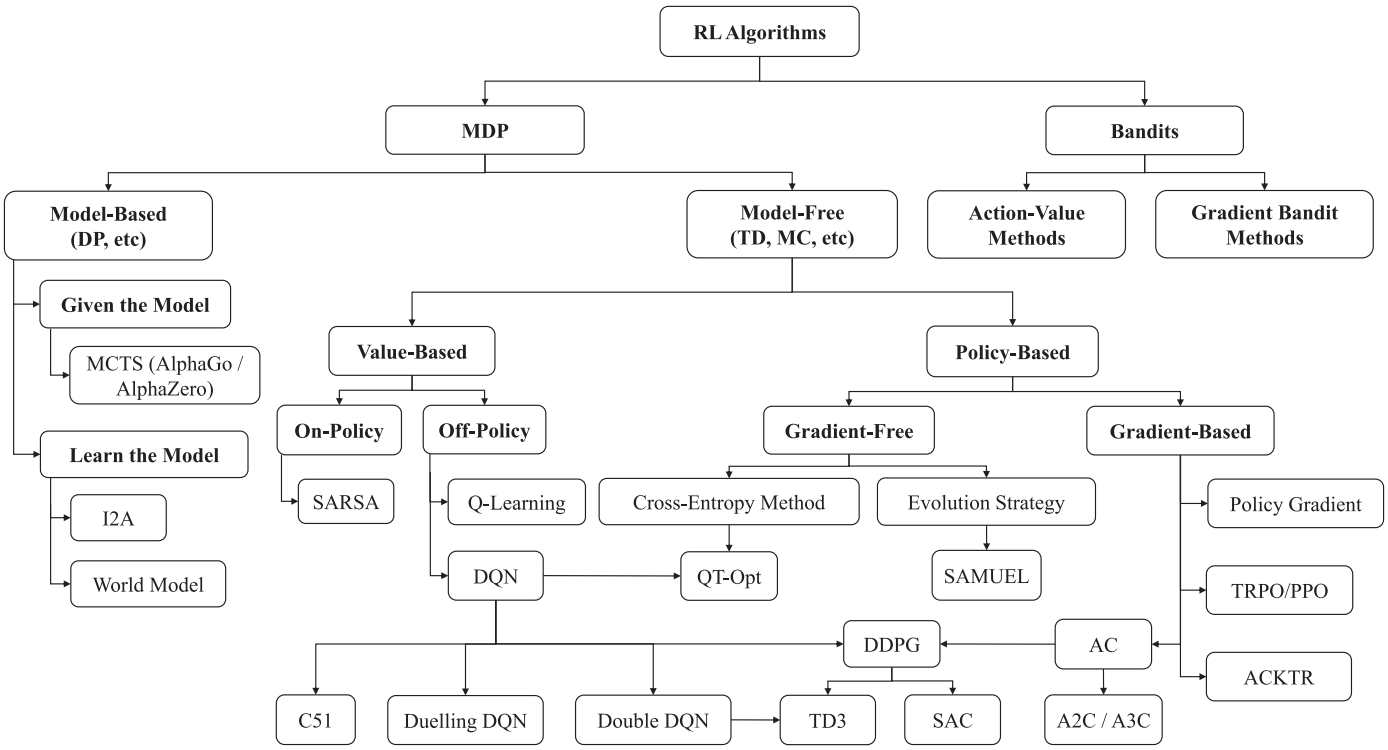

Fig. 2. Taxonomy of the main RL algorithms.
Source: Reprinted by permission from Springer Nature: Zhang and Yu (2020), https://link.springer.com/chapter/10.1007/978-981-15-4095-0_3Taxonomy of Reinforcement Learning Algorithms in Dong, H., Ding, Z., Zhang, S. (eds), https://link.springer.com/book/10.1007/978-981-15-4095-0Deep Reinforcement Learning, Springer, Singapore, pp. 125–133. This Licensed Material is not part of the governing Open Access license but has been reproduced with permission from SNCSC.

Due to the limited resources typically available in end devices, another of the most frequently proposed objectives is the *minimisation of the energy consumption* of the edge computing system. For that aim, the studies typically take into account the energy needed to execute the application, as well as to offload the application and transmit the data necessary for its execution.

Another goal set by many authors is to *maximise the security* of the edge computing system. These articles are largely related to blockchain systems.

In other papers, the authors establish objectives that refer to the use of the devices and the network of the edge computing system. Thus, there are works whose solutions are focused on *maximising the use of network or edge resources*, or *maximising the number of devices that use the edge computing system* by offloading their applications. Other related objectives are the *minimisation of the economic cost* of the system (by setting a price per usage time of the network and edge nodes), and the *maximisation of load balancing in the servers*.

Finally, other authors establish specific objectives for their solutions, such as *maximising the accuracy of the predictions* made by the agent. Another example is the objective of *maximising the reputation of the selected servers*, as they established a reputation system for each of the servers based on the number of tasks executed and discarded.

Tables 3 to 8 again summarise, in their *Objective* column, the specific objectives that are considered in all the papers reviewed in detail in Section 5 of this survey.

3.5. Centralised and distributed decision-making approaches (RQ5)

Depending on where the agents of the RL algorithms are hosted, we have differentiated three approaches that can be found in the literature: centralised, distributed on edge nodes, and distributed on the end devices. These approaches are represented in Fig. 3 and are explained below.

In the *centralised* approach, the system has a single agent, hosted on an edge or cloud server, or on the network controller. This agent is responsible for making offloading decisions for all the end devices on the network. Moreover, this type of agent is characterised by its complete knowledge of the system, including the state of the network, the availability of resources at both servers and end devices, and the set of applications to be executed.

In the *distributed on edge servers* approach, there are multiple agents hosted on edge servers, which are in charge of making offloading decisions for the end devices connected to them. These agents have a limited view of the network, knowing only the resources and applications of the devices connected to the node where the agent is hosted, as well as, occasionally, the resources available on nearby servers. In this approach, collaboration between servers can be considered, where a given server can forward offloaded applications for execution to a nearby server in case of insufficient resources.

Finally, in the *distributed on end-devices* approach, each device has its own agent, which is responsible for determining whether the applications are executed locally or are offloaded. These agents have limited information from the network, knowing only the state of the device resources and communication channels used by the device. However, one of the main challenges of this approach is that the agents are unable to know the decisions made by the agents of other devices and this could congest the communication channels and resources of the edge nodes.

Tables 3 to 8 in Section 5 show, in their *Approach* column, the scenario assumed in each reviewed paper for the placement of the RL agent.

3.6. Number of applications and partitioning (RQ6)

Regarding the applications to be executed, two factors have been taken into account: the number of applications per device, and whether the applications can be partitioned or not.

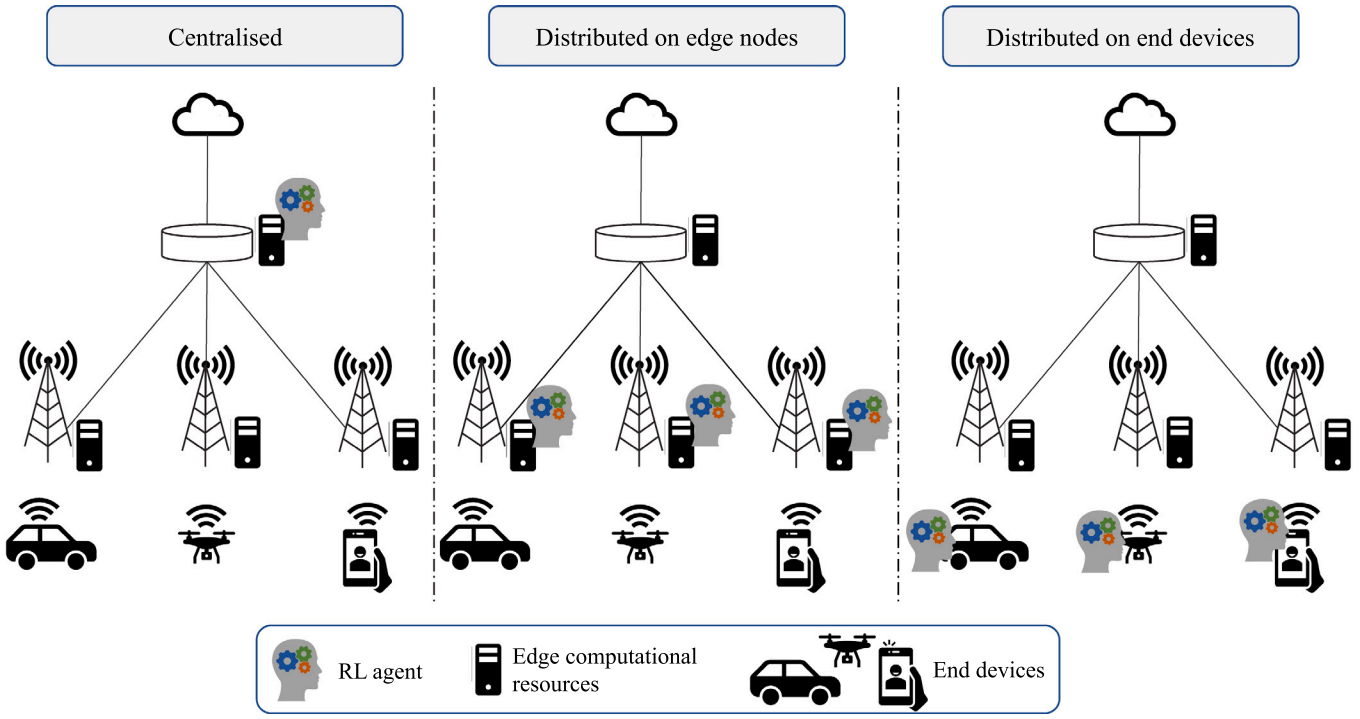

Fig. 3. Different decision-making approaches found in the literature: centralised RL agent (left); RL agent distributed on edge devices (centre); and RL agent distributed on end devices (right).

Regarding the *number of applications per device*, the literature contains studies where a single application per device is considered, while others consider multiple applications per device. Obviously, the latter is a more pragmatic assumption, although it increases the complexity of the system. When multiple applications are assumed, some papers consider that all those applications are simultaneously running on the device from the beginning, while other papers consider a varying number of applications in different time slots.

Regarding the *partitioning of the applications*, two schemes are considered: coarse-grained and fine-grained. The coarse-grained scheme, also known as the binary scheme, means that the application cannot be split. Therefore, it must be entirely executed on the same device (either it is locally executed or offloaded as a whole to another device). In contrast, the fine grained scheme considers that applications can be split. The simplest case of this scheme consists in dividing the execution of the application among several devices ignoring data dependencies, usually representing the division as a percentage of the total load of the application. Other authors consider the partitioning of applications into different tasks, with data dependencies between them (usually considering the application as a Directed Acyclic Graph (DAG), where the nodes represent the tasks composing the application and the links represent the relationships between them), which increases the difficulty of making offloading decisions. Fig. 4 shows these different types of partitioning.

Tables 3 to 8 in Section 5 present, in their *App* column, how applications are handled in the reviewed articles from two points of view. First of all, it is indicated if each end device has a single application to offload or if multiple applications are considered. Secondly, it is indicated if each of these applications can be split into smaller tasks or not, and in the former case, whether inter-task data dependencies are taken into account or not. In these tables, those papers that assume that applications can be split are identified by either a ✓ symbol if data dependencies are considered or a ~ symbol if data dependencies are ignored.

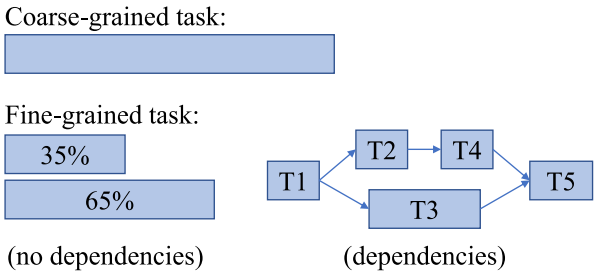

Fig. 4. Differences between coarse-grained and fine-grained tasks, with no dependencies and with dependencies.

3.7. Time-varying aspects (RQ7)

Scenarios analysed in the literature include several time-varying characteristics, namely, communication, energy, application arrival, user movement, server location and server cache.

Probably the aspect most widely considered by the authors as time-varying is related to *communications*. Usually, and due to the type of devices, the end devices of an edge computing system use wireless connections. While these connections have great advantages, they can experience variations in their transmission rate caused by interference, noise or other factors. These variations are considered by most of the papers as they have a significant impact on the offloading decision.

The second aspect considered is the variation of the *energy* level of the end devices with time. For instance, the energy level decreases due to normal operation of the device but may also increase if there is an energy harvesting process. This variation of the energy level is also considered as a parameter to make offloading decisions by many studies, and is particularly important for mobile or IoT devices.

The third aspect taken into account in many papers is the *application arrival* at the end devices. When using RL algorithms, it is common to find articles where they discretise time, dividing it into epochs, slots or

steps. In these cases, the authors usually consider the stochastic arrival of applications, which occurs at the beginning of each time step.

Another aspect considered in this survey is the *device movement*. The computing capabilities of edge systems together with the use of wireless networks make these environments very suitable for mobile devices, vehicles or UAVs, among others. The movement of a user/client device can affect communications by varying the distance between the device and the BS or AP, may require migrations or handovers of offloaded applications.

Movement is also important for servers in vehicular or UAV networks. On these cases, certain vehicles or UAVs can play the role of edge servers, enabling the offloading of applications from other devices. Given the importance of the position of a server when it moves, *server mobility* is an aspect that is also considered in the offloading decision making process in many papers, and thus is also analysed in this review.

The last aspect considered is the edge *server cache*. Some studies consider a cache memory that can store certain applications or services, which can vary over time according to their use. The offloading of applications that are already in the server cache is faster, so the decision to offload a particular application at a given time may vary depending on whether it is in the cache.

Tables 3 to 8 in Section 5, in their *Time-varying Aspects* column, provide a summary of whether the articles reviewed in this survey considered the temporal variation of each of the aspects detailed in this section.

4. Related surveys

This section presents review articles and surveys related to the use of RL-based algorithms to address the problem of computation offloading in edge systems. Moreover, their strong and weak points are given.

Probably the most similar research to the work presented in this article is Shakarami et al. (2020b). In that survey, the authors explore solutions to the computation offloading problem in MEC systems from a machine learning perspective. They included the three main fields of machine learning: supervised learning, unsupervised learning and RL. The main strength of the survey is that it compares different features of each of the mechanisms reviewed: performance metrics, case studies, techniques used, and evaluation tools. However, it only includes articles published up to the beginning of 2020. Only 6% of the articles of the survey are from that year, while most of the articles (77%) were published in 2018 and 2019. In addition, the environment that was considered in each article is not detailed in the survey.

In the same year, Shakarami et al. also presented a survey of articles published between 2016 and 2019 related to computation offloading in MEC environments, from a stochastic perspective, in Shakarami et al. (2020a). This survey focuses on Markov-based stochastic models and analyses a total of 61 articles, although only 23 of them use RL mechanisms.

Like the previous one, several surveys related to computation offloading have been published (Xu et al., 2018; Cao et al., 2019; Masdari and Khezri, 2020; Islam et al., 2021; Saeik et al., 2021). However, they include proposals with different solutions to this problem, with a minority of proposals based on RL. Xu et al. (2018) presented a survey of opportunistic offloading. The concept of opportunistic offloading refers to both offloading traffic transmitted over a cellular network to an opportunistic network, and offloading computing tasks to nearby devices with idle resources over an opportunistic network. Despite the large number of articles reviewed, only two of them use RL techniques. Later, Cao et al. carried out a brief state-of-the-art survey related to the use of artificial intelligence for offloading in MEC (Cao et al., 2019). In this survey they included supervised and unsupervised learning, deep learning, reinforcement learning, and deep reinforcement learning techniques. The main negative point of this review is the low number of articles related to RL, since it only includes 3. Masdari and Khezri (2020) reviewed articles focusing on offloading schemes using Markovian models. Although in this case the authors provide details of the model used and the objective to be achieved, only 13% of the articles in the survey solve the problem by using RL or DRL techniques or mechanisms based on them.

Recently, and in line with the approach described above, Islam et al. also conducted a generic survey on task offloading in MEC systems (Islam et al., 2021). In this case they divided the articles into the algorithms used to solve the offloading issue, namely: Greedy Heuristic, Integer Programming, Machine Learning, Branch and Bound, Dynamic Programming and Convex Optimisation. Given the large number of different algorithms analysed, only 4 articles related to machine learning are included, three of them using DRL techniques. Also within this approach is the survey published by Saeik et al. (2021). It reviews articles focused on task offloading in edge and cloud computing using mathematical, artificial intelligence and control theory solutions. Despite the comprehensiveness of the survey, it is a very generic one, and only includes 5 articles related to RL.

From another perspective, there are surveys or literature reviews that have focused on the use of RL in communications, such as the cases in Zamzam et al. (2019), Qian et al. (2019), Luong et al. (2019). Among these surveys, Zamzam et al. (2019) stands out, where the authors reviewed articles that use machine learning techniques for resource management and computation offloading in MEC. After introducing MEC and its main challenges (cost, energy consumption and latency), this survey presents 13 articles related to computation offloading and resource allocation which use RL techniques. However, this brief survey, which collects articles published between 2017 and 2019, does not refer to the environment considered by the different articles, focusing instead on the provided solution. In the surveys (Qian et al., 2019; Luong et al., 2019) computation offloading is just one of the multiple use cases exposed, and the number of articles referring to it is limited. Qian et al. (2019) reviewed articles related to RL in different fields of communication networks: MEC (including network catching and task offloading), Software-Defined Networking (SDN), network virtualisation and network slicing. This survey is remarkable for the number of covered areas. However, as a weak point, this is a very generic survey, and only includes 4 articles within the area of computation offloading, published between 2016 and 2018. At the same time, Luong et al. carried out a literature review on DRL applications in communications and networking in Luong et al. (2019). After introducing RL and DRL, the authors classify the applications of the articles into four different fields: (1) network access and rate control; (2) caching and offloading; (3) security and connectivity preservation; and (4) miscellaneous issues. Regarding offloading, the articles are divided into different types of offloading models, namely: offloading cellular traffic to Wireless Local Area Network (WLAN), offloading to a single MEC-enabled BS, offloading to a shared MEC server via multiple BSs, and offloading to multiple MEC-enabled BSs and mobile cloudlets. Despite the comprehensiveness of this survey, which includes different areas within communications and networking where RL is applied, only 11.5% of the articles address the issue of computation offloading.

Other surveys collect articles that use RL in specific network environments, where computation offloading is again only one of the different use cases discussed. One of these examples is Mekrache et al. (2021), where the authors reviewed articles that use DRL techniques in vehicular networks, both from a vehicular resource management and from a vehicular infrastructure management perspective. The vehicular resource management includes a section on computation and data offloading. However, it only includes 8 articles related to the application of RL techniques to computation offloading. Another example of this kind of survey is Chen et al. (2021a), where the authors conducted an extensive survey on the use of RL techniques, in this case in the IoT domain. Despite its exhaustiveness and the fact that it includes a multitude of fields that can be considered as part of IoT, the number of articles in each of these fields is limited. For example, this survey

Table 2
Summary of related surveys. No. articles refers only to the number of articles related to the use of RL/DRL for computation offloading in edge computing systems.

Survey	Brief description	No. articles	Weaknesses
Shakarami et al. (2020b)	Machine learning solutions to the problem of computation offloading in MEC, a large part of them based on RL.	41	It only includes articles from 2018 until the very beginning of 2020, with almost 150 articles published afterwards on this topic.
Shakarami et al. (2020a)	Computation offloading in MEC systems.	23	Surveys compiling articles proposing different solutions to the problem of computation offloading in edge computing, although only a few of these solutions are based on RL.
Xu et al. (2018)	Opportunistic Offloading.	2	
Cao et al. (2019)	Artificial Intelligence for offloading in MEC systems.	3	
Masdari and Khezri (2020)	Computation Offloading using Markovian models.	6	
Islam et al. (2021)	Computation Offloading in MEC systems.	3	
Saeik et al. (2021)	Computation Offloading in Cloud and Edge Computing.	5	
Jiang et al. (2019)	Computation Offloading in edge computing systems.	5	
Lin et al. (2020)	Computation Offloading in Edge Computing	6	
Wang et al. (2020h)	Architectures and computation offloading techniques for edge computing.	0	
Mach and Becvar (2017)	Architectures and computation offloading techniques in MEC systems.	0	
Mustafa et al. (2021)	Architectures and computation offloading techniques in MEC systems.	0	
Zamzam et al. (2019)	Use of Machine Learning for Computation offloading, including RL solutions.	13	Surveys with articles using RL to solve different network problems, including computation offloading. However, the number of solution based on RL is limited.
Qian et al. (2019)	Use of RL for different network issues: MEC, SDN, Virtualization and Network slicing.	4	
Luong et al. (2019)	Use of RL in multiple areas: network access and rate control; caching and offloading; security and connectivity preservation; and miscellaneous issues.	15	
Mekrache et al. (2021)	Use of RL in vehicular networks to address different issues from two perspectives: vehicle resource management and vehicular infrastructure management.	8	
Chen et al. (2021a)	Applying RL to the IoT domain.	6	
Chen et al. (2015)	The article includes, in addition to its proposal, a brief survey on communications.	2	
Nomikos et al. (2021)	Use of RL for caching in edge and MEC systems.	5	
Althamary et al. (2019)	Study of RL techniques in vehicular networks.	2	

only includes 6 articles in the area of computation offloading. Chen et al. (2015) also included, along with their proposal, a brief review of the literature. In this review they classify the papers according to communications (through small cells, through WiFi networks and through opportunistic communications). However, given the briefness of the section dedicated to the literature review, only a couple of articles use RL techniques to solve the offloading problem.

In addition to all the surveys and literature reviews presented in this section so far, other surveys have also been published that marginally include some articles that use RL techniques to solve the computation offloading issue. An example of such surveys is Nomikos et al. (2021), where the authors focused on articles using RL for caching in edge and MEC systems. Another example is Althamary et al. (2019), where the authors reviewed articles that use RL methods in vehicular networks. However, the number of articles included within that study related to the field of computation offloading is only two. In contrast, Jiang et al. (2019) focused their survey on computation offloading in edge computing. Although this survey does not provide a section for RL-related articles, 5 of the surveyed articles use RL or RL-based techniques. Similarly, Lin et al. (2020) conducted a survey on computation offloading for edge computing. They divided the surveyed articles according to the method used to solve the computation offloading problem. Among the different sections they include machine learning and RL, but only 6 articles related to RL were considered. Finally, other surveys, such as Mach and Becvar (2017), Wang et al. (2020h) and Mustafa et al. (2021), analyse architectures and offloading computation techniques in MEC, but do not include RL approaches.

Table 2 summarises the surveys related to our work, including a brief description of the survey, the number of articles from each of them with RL-based solutions to deal with the management of computation offloading and the main weakness.

Finally, we also present a summary of the need for this survey on the use of RL mechanisms for computation offloading in edge computing systems, based on the weaknesses found in the surveys and reviews included in this section.

- The closest surveys to our focus, review articles published up to 2019 (and early 2020), leaving a large number of recent articles excluded from the study.
- Some surveys present RL-based solutions for the computation offloading issue within a more general context. This means that, despite the large number of articles they reviewed, the number of articles included that use RL mechanisms is limited.
- Some surveys focus on RL mechanisms, but applied to a broad target (such as communication and networking in general), resulting in a limited number of articles applying RL to computation offloading.
- The number of articles reviewed in some surveys is not representative given the number of publications in the field.
- Some surveys do not detail the methodology followed in the literature search.

5. Recent advances on reinforcement learning for computation offloading

This section reviews and classifies recent work on RL for computation offloading. The papers have been selected for inclusion following the steps described in Section 2.

Due to the large number of articles to be reviewed, it has been found convenient to divide them for a better analysis, in order to find possible trends or preferences when considering environments and proposing solutions. At this point, it has been considered appropriate to divide them according to their case studies, as usually the objectives and metrics tend to differ between case studies, but remain similar within each case study. Thus, the different case studies we have encountered are as follows:

- IoT networks: this type of networks usually consists of multiple end devices with low power consumption, so reducing energy consumption is often one of the main objectives of the articles in this case study.

Table 3
Summary of literature related to the IoT case study (LAT: Minimise Application Latency; EN: Minimise Energy Consumption; CT: Maximise Completed Tasks; CR: Maximise Computation Rate). Regarding App. partitioning: ∼, allowed, but data dependencies ignored; √, allowed and data dependencies considered.

Article	Algorithm	Objective	Approach			App.		Net. Architecture				Time-varying Aspects					
			Centralised	Distributed (End Devices)	Distributed (Edge Servers)	Multi-Application	Application Partitioning	Multi-Device	Multiple BSs	Multiple Edge Servers	Cloud	Communication	Energy	Application Arrival	Device Movement	Server Location	Server Cache
Xu et al. (2020)	QL	LAT, EN	✓			✓		✓	✓	✓	✓			✓			
Cui et al. (2020)	QL	LAT, EN		✓				✓				✓					
Liu et al. (2020d)	QL	LAT, EN		✓		✓		✓				✓		✓			
Liu et al. (2020e)	DQN	LAT, EN		✓		✓		✓	✓	✓	✓	✓		✓			
Gong et al. (2020)	DQN	LAT, EN	✓			✓		✓									
Khan et al. (2020)	DQN	EN	✓			✓		✓			✓						
Huang et al. (2020a)	DRL-based	CR	✓					✓				✓					
Qian et al. (2021)	DRL-based	EN		✓	✓	✓	∼		✓	✓		✓					
Long et al. (2020)	A3C	LAT, EN	✓			✓		✓	✓	✓	✓			✓			
Shu et al. (2021)	A3C	EN, CT		✓		✓		✓	✓	✓	✓						
Li et al. (2021c)	AC	LAT, EN	✓			✓		✓						✓			
Zhang et al. (2020b)	AC	LAT, EN		✓		✓	∼	✓						✓			
Zhang et al. (2020d)	AC, DDPG	LAT, EN		✓		✓	∼	✓	✓	✓				✓			
Zhang et al. (2020c)	AC, DDPG	LAT, EN	✓	✓		✓	∼	✓	✓	✓			✓	✓			
Deng et al. (2021)	QL, DDPG	LAT	✓			✓	∼	✓				✓					
Li et al. (2020d)	DDPG	LAT, CT			✓	✓	∼	✓	✓	✓	✓			✓	✓		
Liu et al. (2020a)	DDPG	LAT	✓			✓		✓				✓		✓			
Ale et al. (2021a)	Dirichlet DDPG	LAT, EN	✓			✓	∼	✓						✓			
Lu et al. (2020)	Double-Dueling DPG	CT, EN	✓			✓	∼	✓	✓	✓				✓			

- Vehicular networks: these case studies are often characterised by the movement of their end devices. This results in a great number of articles considering device movement as a time-varying aspect and, associated with this, the communication between devices.
- Use of UAVs: as in the previous case, this type of network is largely related to the movement of devices. However, in this case, UAVs are often used as mobile BSs or APs and to provide computational resources to nearby devices. Therefore, it is usual that in papers whose proposal is related to UAVs the authors consider the location of the edge servers to be time-varying.
- Specific use cases: this case study covers articles which used edge networks for a very specific application, not commonly considered, such as virtual reality or robotics.
- Generic studies: this case study includes generic studies, where the authors consider generic networks with devices described as mobile devices or user devices, for example.

All these articles have been classified according to different criteria presented in Section 2, and that classification is shown in the following subsections in a set of accompanying tables

5.1. IoT

This subsection presents articles related to case studies of networks with edge computing systems where the end devices are related to the IoT field. A summary with the main features of those works is shown in Table 3.

One of the most commonly used approaches in the analysed articles to solve the problem of computation offloading in IoT systems is Q-learning (from now on, QL). For example, Xu et al. (2020) proposed a mechanism based on RL, the Lyapunov optimisation theory and the Lagrange theorem to solve the communication and computational resource allocation problem in a cloud–edge-terminal network, in order to optimise the utilisation of edge resources and terminal energy. Another example can be found in Cui et al. (2020), where the authors studied the computation offloading problem in a dynamic multi-user environment, where wireless channels could experience interference when used by multiple devices. To solve this problem, the authors first proposed a multi-user computation offloading method based on evolutionary game theory. However, due to the dynamic nature of devices and communication channels in practical environments, the devices had problems understanding the strategies of others, so the authors then proposed an evolutionary game algorithm based on QL. The main limitation of this paper is that, although the authors considered time-varying communications, there were no other dynamic aspects, and the scenario was very simple, with a single BS and an edge server. Liu et al. (2020d) also proposed a QL based solution to the computation offloading problem, but from a selfish approach. They also considered a system with a single edge server connected to a BS and multiple IoT devices, each with a learning agent that observed its local environment and took decisions on whether to offload or not. The main problem in this situation was the possibility of other users offloading their tasks at the same time, coupling the gateway. Simulations showed that the proposed algorithm solved the problem in a more energy-efficient way than a centralised approach.

Most of the articles reviewed apply DRL techniques. Among the value-based DRL techniques, DQN is particularly relevant. This technique was used by Liu et al. (2020e). In this case, their approach included two steps: firstly, devices were centrally clustered according to the priority to offload their tasks; secondly, a distributed DQN-based computation offloading algorithm optimally selected the tasks to be offloaded by each device. This is not the only solution to the problem of computation offloading in IoT scenarios that is based on DQN, finding several examples among the reviewed articles. For instance, Gong et al. (2020) introduced the MEC service to an IoT system to leverage computation offloading and resource allocation in different applications executed by IoT devices. In this paper, they proposed a decision-making algorithm based on DQN to optimise task offloading and resource allocation. Another example of the use of DQN can be found in Khan et al. (2020), where the authors focused on Machine Type Communication Devices (MTCDs), proposing a MEC system with a edge server and a cloud server. Due to the lack of computational power, MTCDs sent their tasks to the edge server, which could send them to the cloud for execution in case its processing units were overloaded.

Other articles do not refer to the use of DQN algorithms as such, but propose the use of DQN-based algorithms or their proposals are similar to the behaviour of the DQN algorithm. Huang et al. (2020a) proposed a DRL-based Online Offloading (DROO) framework that takes offloading and resource allocation decisions optimally in a MEC system whose wireless communications vary over time. This framework implemented a DNN as a scalable solution suitable especially for large networks. In addition, to further reduce complexity, the authors proposed an adaptive procedure that adjusts the parameters of the algorithm on the fly. However, the authors consider a relatively simple scenario (they consider multiple end devices with only one BS equipped with an edge server) and no dynamism. Moreover, although the authors state that their algorithm converges quickly and gives a solution in a shorter time than other methods, their article does not present a comparison with other algorithms. There are other examples of DRL with DNN-based solutions, such as Qian et al. (2021), who considered a time-varying channel Non-Orthogonal Multiple Access (NOMA) assisted multi-server MEC scenario with a single IoT device with multiple tasks. In this scenario, the authors proposed a distributed DNN-based algorithm to solve the optimisation problem of the computation offloading, NOMA transmission and computation resource allocation with the objective of minimising the energy consumption of the IoT device. The proposed algorithm decomposed the problem so that it could be solved distributed among the IoT device and the edge-computing servers.

Another approach of RL techniques is the use of gradient-based methods, where AC and its variants stand out. Among the articles reviewed we find Long et al. (2020), where the authors proposed an offloading scheme for an IoT-edge-Cloud network. Due to the low cost and low power consumption requirements of IoT devices, they are usually not equipped with communication modules to offload tasks to MEC or cloud servers. Therefore, the authors proposed to use vehicles in a smart city as network nodes to which IoT devices could offload their applications. The vehicles could then execute these applications or offload them to the MEC or cloud servers. In any case, the results obtained from the execution of these applications were collected by the Cloud server to make scientific decisions. Once a vehicle received the task, the A3C algorithm, an AC variant, was used to choose where it was executed, taking into account the delay of the tasks and the energy consumption. Another article that also uses the A3C algorithm is Shu et al. (2021), where the authors proposed a solution for computation offloading in a cloud–edge-terminal system. However, unlike other articles, the A3C algorithm proposed by the authors did not take the offloading decision directly, but selected the algorithm to be used to make the decision, choosing between a greedy algorithm and Particle Swarm Optimisation (PSO). This allows for better results compared to using either algorithm separately, without the computational overhead of a DRL algorithm making the final offloading decision. There are also multiple solutions based on the AC algorithm itself, such as Li et al. (2021c), who considered a NOMA MEC system with multiple IoT devices, in which the IoT devices could cooperate with each other using short range communications to assist them offloading computation to the edge servers. Here, authors proposed a hierarchical multiagent DRL framework where the agents, based on AC algorithm, were within a league to explore the environment in a collaborative way. Moreover, authors trained the intelligent agent using expert strategies in order to accelerate convergence.

Other authors using AC-based methods were Zhang et al. who published three papers related to the topic of this survey (Zhang et al., 2020b,d,c). Although the study case in the two first articles is not an IoT environment, we found it convenient to include it here due to the consideration of energy harvesting of the devices, as well as the similarity with their third article, which extends the work started in the first two articles. In Zhang et al. (2020b), the authors considered a MEC system with multiple mobile devices with energy harvesting and a MEC server. In this environment, the devices decided how much of the tasks they offloaded and the portion of their computational capacity (which was energy dependent) they used for local executions. To reduce the execution time and energy consumption, the authors proposed a DRL algorithm based on AC, with a centralised policy for learning that could then be executed on each device. Afterwards, Zhang et al. also proposed a multi-server environment with one user in Zhang et al. (2020d), where variable computing tasks and changing computation capacity of servers made the problem of computation offloading challenging. In this scenario, the authors proposed the use of a continuous–discrete hybrid decision based AC algorithm. Finally, the authors extended this work in Zhang et al. (2020c), where they focused on the study of a multi-device multi-server MEC system for IoT devices with energy harvesting. In this article, they also considered a dynamic system, where the energy of the devices and the computing capabilities of the servers varied due to possible requests from other devices. To address this problem, the authors proposed two DRL-based algorithms: (1) hybrid-decision-based AC learning and (2) multi-agent DDPG for dynamic computation offloading. The first algorithm selected the server for offloading in a centralised manner, while the second adopted the framework from the centralised training to a decentralised execution. Another article that proposes two different solutions, one of them being DDPG is Deng et al. (2021). In this paper, the authors studied the problem of partial computation offloading in Industrial Internet of Things (IIoT) MEC systems, where IIoT can be defined as the extension of IoT to the industrial sector, and which is considered one of the pillars of Industry 4.0. Here, the authors considered a multi-user scenario with a single MEC server, where they proposed the use of two different algorithms: QL and DDPG. In this case the algorithms also had to decide which part of the applications was executed locally and which part was offloaded to the MEC server, extending the actions to the continuous action domain. Therefore, the algorithm with the best solution was DDPG, despite its slower convergence speed.

The DDPG algorithm used in the last articles combines ideas from the AC algorithm and the DQN algorithm discussed above, and has also been used in several papers in this study case. One of these papers is Li et al. (2020d), where the authors studied the computation offloading problem in a scenario with three network layers: IoT devices, including static and mobile devices, heterogeneous edge servers, and cloud servers. In this scenario, the authors proposed a distributed algorithm based on DDPG with the objective of minimising application latency and maximising task completion. Liu et al. (2020a) also used a DDPG-based computation offloading method to minimise latency in a Wireless Power Transfer (WPT) network with a MEC server and several IoT devices. These IoT devices were charged by the MEC server, using that energy either to process the data locally or to send it to the MEC server. Ale et al. (2021a) considered a MEC system with multiple users and multiple edge servers. In this scenario, the authors also proposed an algorithm based on DDPG, called Dirichlet Deep Deterministic Policy Gradient, since it adopts the Dirichlet distribution. This algorithm was responsible not only for taking offloading decisions, but also for determining the partitioning of the applications into sub-tasks and their distribution on the edge servers for their execution, which could be parallel. Another example is Lu et al. (2020), where the authors investigated the computation offloading problem in order to improve Quality of Experience (QoE) in an edge IoT network. To improve QoE, the authors aim to reduce the latency of applications and increase their execution rate, while reducing energy consumption. To achieve this, they proposed a Double-Dueling-Deterministic Policy Gradients algorithm based on DDPG, which improved instability and the slow convergence of the DDPG algorithm. However, as in the previous article, only the arrival of applications is considered to be time variant.

According to Table 3, this type of use case does not reveal a clear use of one type of algorithm, finding solutions based on a wide range of RL algorithms. The minimisation of energy consumption figures prominently in the objectives considered by the authors, due to the type of systems proposed, where it is of great importance. Another important

Table 4
Summary of literature related to the Vehicle Case Study (COST: Minimise Economic Cost; LAT: Minimise Application Latency; EN: Minimise Energy Consumption; TR: Maximise Transmission Reliability; SEC: Maximise System Security; NET: Minimise the use of network resources; ER: Minimise the use of Edge Resources (computation, communication and/or storage); CT: Maximise Completed Tasks; BA: Load Balancing; DEV: Maximise the number of devices which uses edge resources). Regarding App. partitioning: ∼, allowed, but data dependencies ignored; √, allowed and data dependencies considered.

Article	Algorithm	Objective	Approach			App.		Net. Architecture				Time-varying Aspects					
			Centralised	Distributed (End Devices)	Distributed (Edge Servers)	Multi-Application	Application Partitioning	Multi-Device	Multiple BSs	Multiple Edge Servers	Cloud	Communication	Energy	Task Arrival	Device Movement	Server Location	Server Cache
Li and Xu (2021)	QL	COST	✓			✓		✓	✓	✓	✓	✓			✓		
Cui et al. (2021)	QL	LAT, TR	✓			✓	∼	✓	✓	✓	✓	✓					
Tang et al. (2020)	DQN	LAT, EN		✓					✓	✓		✓			✓	✓	
Chen et al. (2021f)	DQN	LAT		✓			∼	✓	✓	✓		✓			✓	✓	
Wu and Yan (2021)	DQN	LAT, EN	✓			✓		✓	✓	✓	✓	✓			✓		
Zheng et al. (2021)	DDQN, Dueling DQN	LAT, EN, SEC	✓			✓		✓			✓	✓			✓		
Luo et al. (2020)	DQN	EN, COST	✓			✓		✓	✓	✓				✓	✓		
Liu et al. (2020b)	DQN	LAT		✓		✓			✓	✓		✓		✓	✓		
Khayyat et al. (2020)	DQN	LAT, EN	✓					✓	✓	✓	✓				✓		
Wang et al. (2020b)	DDQN	LAT, EN	✓			✓			✓	✓	✓	✓		✓	✓		✓
Zhang et al. (2020e)	DQN	NET		✓					✓	✓					✓	✓	
Zhan et al. (2020b)	PPO	LAT, EN	✓			✓						✓		✓	✓		
Liu et al. (2020f)	PG	LAT	✓			✓	✓	✓	✓	✓	✓	✓					
Liu et al. (2021d)	PG	LAT, EN		✓			✓	✓	✓	✓		✓					
Zhu et al. (2021)	AC	LAT	✓	✓		✓		✓	✓	✓		✓		✓	✓		
Xu et al. (2021)	AC	SEC	✓	✓		✓	∼	✓	✓	✓	✓			✓	✓	✓	
Wang et al. (2020d)	A3C	LAT	✓			✓	∼	✓	✓	✓				✓	✓		
Ke et al. (2020)	DDPG	LAT, EN, ER	✓			✓		✓	✓			✓		✓			
Li et al. (2020b)	DDPG	LAT, CT	✓			✓	∼	✓	✓	✓	✓	✓		✓	✓		
Geng et al. (2021)	DDPG	LAT, EN, BA		✓		✓	∼	✓		✓		✓		✓	✓		
Zhang et al. (2020a)	DDPG	DEV	✓			✓		✓	✓	✓				✓	✓		✓
Huang et al. (2020b)	DDPG	EN		✓		✓		✓	✓	✓		✓			✓		
Shi et al. (2020)	DDPG	EN, ER, NET	✓			✓			✓	✓	✓	✓			✓		

point to note is that a large number of articles have a centralised approach, where a single agent is responsible for making decisions for multiple nodes. In addition, except for specific cases, these articles consider networks with multiple end devices with multiple tasks.

5.2. Vehicular networks

This subsection includes articles related to the case study of vehicular networks in edge computing systems. Table 4 classifies the articles related to this type of networks.

Starting with a classical RL algorithm, Li and Xu (2021) proposed a QL algorithm to address the problem of computation offloading and resource allocation in a network with vehicles, Roadside Units (RSUs) equipped with MEC servers and Cloud servers, in order to minimise the cost (taking into account the cost of computational resources and communication resources). Another example is Cui et al. (2021), in which the authors considered a scenario with multiple vehicles, RSUs with MEC servers and a cloud server, where vehicles produced applications that could be executed locally or offloaded to other vehicles, or to the MEC or Cloud servers. In addition, the authors considered application partitioning, but only in the same layer. For this, they proposed a system with three algorithms. First, a K-Nearest Neighbour (KNN) algorithm chose in which layer a vehicle application should be executed. In case of selecting the offloading to another vehicle, another non-RL based algorithm was used to select the vehicle taking into account parameters such as the distance or the speed of the vehicles. Finally, in case of selecting to offload the application to the MEC layer, a QL algorithm was used to select the server or servers where to offload the application. The main negative point of this article is that the authors did not consider the movement of vehicles as a time-varying aspect, despite the fact that it was a vehicular network.

In computation offloading in vehicular networks, DQN approaches are among the most widely used. An example of this is Tang et al. (2020), where the authors considered a vehicular network consisting of a vehicle with several applications, which have to be executed in sequential order, and multiple vehicles that could be used to offload the tasks. In order to minimise delay and energy consumption, the authors proposed a DQN-based algorithm. Chen et al. (2021f) also proposed a distributed offloading computation strategy for IoV based on DQN. In addition, the authors considered a scenario where there are no (or insufficient) MEC servers as such. As a solution, they proposed to use the unused resources of vehicles to execute tasks of the requesting vehicles, with communication being performed directly between the different vehicles.

However, it is unusual to find networks where the edge servers are only the vehicles themselves, but are usually located in the RSUs or APs. For example, Wu and Yan (2021) considered a MEC system with multiple RSUs with computing resources (edge servers) and a cloud server, where multiple moving vehicles could offload their applications for execution. In addition, due to the movement of vehicles, the authors addressed application migration. In this scenario, the authors proposed a DQN-based algorithm to reduce latency and energy consumption. Zheng et al. (2021) investigated the computation offloading and the access control problem. In this case, the authors also considered a scenario with multiple vehicles that could offload their tasks to a edge or Cloud server. Moreover, to improve the security of the system, the authors proposed the use of blockchain and smart contracts for access control. Then, the authors proposed a computation offloading scheme based on DDQN and Dueling DQN for optimal offloading and resource and bandwidth allocation decisions. Another example is Luo et al. (2020), where the authors studied the computation offloading problem in a vehicular network with multiple moving vehicles and RSUs. In the considered scenario, data could be processed locally, offloaded to RSUs,

migrated to collaborative vehicles or cached in queues. As a solution, they proposed a DQN-based algorithm to optimise data scheduling. Liu et al. (2020b) also considered a scenario in which a vehicle drove through a route on which there were multiple edge servers. In order to select the best server on which to offload its tasks, they proposed the use of two algorithms: a first offline algorithm to select the best candidates before starting the drive and a second online DQN-based algorithm to select the optimal server from the candidate set based on the network conditions.

Furthermore, Khayyat et al. (2020) considered a scenario with a vehicular network with multiple edge servers and a cloud server, in which they studied the problem of computation offloading and resource allocation. To obtain a near-optimal solution, the authors proposed a DQN algorithm that used multiple deep neural networks in parallel. Another example is Wang et al. (2020b), where the authors presented a MEC system for vehicular networks, in which they contemplated several MEC servers and a Cloud server. Among the considerations taken by the authors, the use of cache in the MEC servers that certain tasks may require for their execution stood out, making a migration necessary in case the MEC server selected to execute the task did not have the corresponding cache. In this system, the authors proposed a DDQN algorithm. Other authors also considered the possibility of mobile servers, such as Zhang et al. (2020e), who focused on vehicular networks and the problem of frequent handovers in such networks due to movement. To deal with this problem, the authors designed a novel follow-up computation offloading paradigm, where moving servers can provide extra computing resources. They proposed the use of a DQN as a computation offloading strategy to improve the QoS.

Regarding the use of gradient-based RL methods, we find different proposals, such as Proximal Policy Optimisation (PPO) (Zhan et al., 2020b), PG (Liu et al., 2020f, 2021d), AC (Zhu et al., 2021; Xu et al., 2021) or A3C (Wang et al., 2020d). Zhan et al. (2020b) considered a dynamic wireless environment with stochastic task generation. In this scenario with moving vehicles, the authors proposed a PPO-based algorithm to make the decision of when and where to offload in order to minimise latency and energy consumption. However, although the authors described a scenario with a network architecture with multiple devices in all layers, the action space of the RL algorithm was more limited, taking into account only the device that generates the task, a BS and a MEC server. On the other hand, Liu et al. (2020f) proposed an offloading scheme based on PG to minimise the latency of applications in the field of Internet of Vehicles (IoV). In particular, the authors considered a multi-vehicle environment with multiple edge servers. An interesting aspect of this paper is the division of the applications into tasks with dependencies represented as DAGs. However, as a drawback, the authors only compare their proposal with non-RL algorithms. The authors continued their work in Liu et al. (2021d), where they considered a vehicular network in a MEC environment, with multiple vehicles and multiple MEC servers connected to different RSUs, as well as a central MEC server. In addition, the authors also considered the division of applications into tasks with dependencies, represented by a DAG, using a PG-based algorithm for computation offloading. However, in this case the algorithm had a distributed approach, with an agent in each vehicle, which made decisions based on the tasks of the vehicle, the known state of the network and the past decisions of other vehicles. Moreover, this time the authors compared their proposal with two RL algorithms (AC and DQN), as well as with other offloading schemes showing the higher convergence and better performance of their proposal. Despite being a vehicular network, in neither of the two articles did the authors refer to the movement of vehicles.

Concerning proposals based on the AC and A3C algorithms, Zhu et al. (2021) proposed a multi-agent deep reinforcement computation offloading (MDRCO) scheme. The authors considered a vehicular edge network with a data centre, MEC servers, and vehicles. The data centre was responsible for the training model, while each vehicle contained an agent which selected a MEC server to offload tasks. The multiagent DRL algorithm proposed by the authors trained the actor network and critic network in a centralised way, making real-time task offloading decisions in a distributed way. Xu et al. (2021) focused on a secure computation offloading scheme for vehicular networks based on blockchain. For that, the authors proposed a scenario with multiple vehicles (being some of them resource requesters and the others resource providers), multiple RSUs with computing resources and a BS with computing resources and connected with the remote computing centre. In this scenario, authors considered some malicious vehicles (both resource requesters and resource providers). The proposed scheme comprised the blockchain based trust management and an AC algorithm based smart contract, providing a secure and intelligent computation offloading scheme. This algorithm had a centralised training and a decentralised execution in order to guarantee its convergence and security. Wang et al. (2020d) investigated the use of partial computation offloading in vehicular networks. In this type of networks, they proposed two scenarios, one single-vehicle and one multi-vehicle, where vehicles could partially offload their tasks for execution on MEC servers using the A3C algorithm.

Another of the most commonly used solutions in the articles was DDPG, which is proposed in Ke et al. (2020), Li et al. (2020b), Geng et al. (2021), Zhang et al. (2020a), Huang et al. (2020b), Shi et al. (2020). Ke et al. (2020) proposed an adaptive computation offloading method based on DRL that dealt with the offloading computation problem for MEC in vehicular networks. This algorithm was based on DDPG, and took into account multiple stochastic tasks, the variety of wireless channels and bandwidth, although the authors did not consider the movement of the vehicles as a dynamic aspect, and a single MEC server was considered. Other authors instead considered multiple edge servers, such as Li et al. (2020b), who developed a collaborative edge computing framework to reduce the latency and improve service reliability for vehicular networks. For this purpose, they proposed firstly a task partition and scheduling algorithm to decide the load location and execution order of tasks, and secondly a DDPG algorithm to determine the task offloading, computing and result delivery policy for vehicles. Geng et al. (2021) also considered a vehicular MEC network with multiple servers. This MEC environment included multiple vehicles connected to a BS. The authors proposed the use of a DDPG-based algorithm for computation offloading to minimise the latency and energy consumed in the execution of the applications, as well as load balancing between local and server execution. The authors gave this algorithm a distributed approach, as in their articles (Liu et al., 2020f, 2021d), with an agent in each vehicle making decisions based on the applications of the vehicle and the state of the network. Another example is Zhang et al. (2020a), who developed a social-aware edge computing and caching mechanism by exploiting the relation between vehicles and RSUs. Using DDPG, authors proposed optimal content processing and caching schemes in order to maximise the dispatch utility in vehicular networks. Huang et al. (2020b) also considered a scenario with multiple RSUs equipped with computing resources and different vehicles. In this scenario, the authors proposed a computation offloading and resource allocation algorithm based on a multi-agent DDPG, which considered the speed of the vehicles. Finally, some papers proposed a multi-tier network, including edge and cloud servers. One of these articles is Shi et al. (2020), where the authors proposed a mobility-aware computation offloading method in a MEC system with a vehicular network. This MEC system included two processing levels, which allowed MEC servers to send their raw tasks to the core network for execution. In order to learn the optimal offloading policy, the authors proposed the use of a DDPG-based algorithm.

In vehicular networks, the number of variables and their dimensionality is greater than in other environments where the devices are static or have limited movement. As Table 4 shows, despite the different options, DRL-based solutions predominate. In this case, one of the most considered objectives is the minimisation of application latency, due to the multitude of real-time and minimum latency applications that

Table 5
Summary of literature related to the UAV Case Study (LAT: Minimise Application Latency; EN: Minimise Energy Consumption; ER: Minimise the use of Edge Resources (computation, communication and/or storage); PA: Prediction Accuracy; NET: Minimise the use of network resources; CR: Maximise Computation Rate; CT: Maximise Completed Tasks; SEC: Maximise System Security; QoS: Maximise user QoS). Regarding App. partitioning: ~, allowed, but data dependencies ignored; √, allowed and data dependencies considered.

Article	Algorithm	Objective	Approach			App.		Net. Architecture				Time-varying Aspects					
			Centralised	Distributed (End Devices)	Distributed (Edge Servers)	Multi-Application	Application Partitioning	Multi-Device	Multiple BSs	Multiple Edge Servers	Cloud	Communication	Energy	Task Arrival	Device Movement	Server Location	Server Cache
Kim et al. (2020)	QL	LAT, EN	✓			✓		✓	✓	✓	✓	✓		✓		✓	
Wang et al. (2020f)	QL	LAT, ER			✓	✓		✓		✓						✓	
Wang et al. (2020e)	QL	LAT, EN			✓	✓		✓		✓	✓			✓	✓	✓	
Shi et al. (2021)	DQN	EN		✓		✓		✓	✓	✓	✓	✓		✓	✓	✓	
Qu et al. (2021)	QL	PA, LAT, EN	✓			✓	~	✓	✓	✓				✓	✓	✓	
Liu et al. (2020c)	DQN	NET, CR	✓		✓	✓		✓	✓	✓				✓			
Sha and Zhao (2021)	DQN	LAT, EN	✓			✓		✓	✓	✓		✓					
Ren et al. (2021)	DQN, DDPG	LAT			✓	✓		✓	✓	✓		✓		✓	✓	✓	
Zhang et al. (2021e)	DQN	EN, CT	✓			✓	~	✓				✓		✓		✓	
Wang et al. (2020c)	DDQN	LAT, EN, NET	✓			✓		✓		✓		✓	✓	✓		✓	
Ke et al. (2021a)	DQN	LAT, EN		✓		✓		✓	✓	✓		✓	✓	✓	✓	✓	
Zhu et al. (2020)	A2C	LAT	✓			✓	✓	✓		✓		✓		✓		✓	
Wang et al. (2021)	DDPG	LAT	✓			✓	~	✓				✓	✓	✓		✓	
Wei et al. (2021)	DDPG	LAT, EN		✓		✓		✓	✓	✓				✓		✓	
Dai et al. (2021)	DDPG	EN	✓			✓		✓	✓	✓		✓		✓	✓	✓	
Seid et al. (2021b)	DDPG	LAT, EN	✓	✓		✓		✓	✓	✓		✓		✓		✓	
Seid et al. (2021a)	DDPG	LAT, EN	✓		✓	✓	~	✓	✓	✓		✓		✓		✓	
Li et al. (2021b)	SAC	LAT, EN, CT	✓			✓	~	✓		✓		✓		✓		✓	
Mohammed et al. (2020)	DRL-based	SEC, QoS				✓		✓	✓	✓	✓						

are related to this particular networks, such as autonomous driving or accident avoidance applications.

There is also a larger number of papers where the proposed algorithm has a centralised approach, where a single agent takes the computation offloading (and resource allocation, if applicable) decisions. In these scenarios, where users move around the network (and can even move in and out of the network), it is more difficult for agents to learn if they only have local information from the end devices, so it is common to see centralised approaches or, in some cases, distributed approaches with centralised training. Of course, these types of environments are notable for the movement of their vehicles or end-devices, being very specific cases those that do not consider this.

5.3. UAVs

This subsection includes articles related to the use case of networks including UAVs as end devices or, more commonly, as mobile edge servers which enables the execution of applications offloaded from end-devices, increasing the adaptability of the network. Articles related to the use of UAVs are classified in Table 5.

QL-based solutions are notable in this case study, where we find articles such as Kim et al. (2020), where the authors proposed a QL-based method for computation offloading in a UAV-assisted MEC environment, enabling UAVs to execute tasks offloaded from other devices within their coverage range. In this environment, the authors took into account the position and distance of the devices when calculating power consumption and execution delay. Wang et al. (2020f) also proposed the use of UAVs to provide offloading computation opportunities to mobile users in a MEC system, with UAVs playing the role of edge servers using a QL algorithm. Due to the hardware limitations of UAVs, the objective is to minimise the latency of execution of user device tasks while minimising the resource consumption of UAVs. Wang et al. (2020e) extended their system to include the cloud platform and detailed in depth their proposal. Then, some of these authors, in Shi et al. (2021), also addressed the problem of computation offloading in scenarios with multiple devices, multiple UAVs and a cloud layer. In this case, they focused on IIoT scenarios, and proposed a DQN-based algorithm where the neural network was trained with prioritised samples in experience replay. In turn, Qu et al. (2021) focused on a multi-drone-edge video analytic network, in which drones could offload their tasks to other drones or to an edge or cloud server if they had low battery or required more computing resources. In this environment, the authors proposed a framework, called DroneCOCoNet, for computation offloading and network control using two different approaches, a heuristic algorithm and a QL-based algorithm.

There are also articles that proposed solutions based on DRL. We first describe articles that use DQN algorithms, or algorithms based on them, to solve the problem of computation offloading in networks with UAVs. Thus, Liu et al. (2020c) focused on the use of UAVs to provide computing services in remote areas. To solve this problem, they first proposed a cooperative MEC system that operates through UAVs, which help other UAVs to execute the tasks offloaded by the devices using computation services if necessary. Then, the authors proposed a cooperative computation offloading scheme based on DQN taking into account the randomness of the tasks required by the devices and the situation of the communication channels from two approaches: centralised and distributed. Sha and Zhao (2021) also considered a multi-user MEC system with multiple MEC servers located in UAVs. However, the authors considered neither the mobility of the end devices nor the mobility of the UAVs. In this scenario, the authors also proposed a DQN algorithm to take computation offloading and resource allocation decisions, in this case to minimise application processing delay and energy consumption. Ren et al. (2021) investigated a multi-user UAV-assisted MEC network, but in this case the authors focused on a large-scale multi-UAV network. In order to minimise the task delay of all mobile devices, the authors proposed a hierarchical RL called HT3O. This approach decomposed the problem into two sub-problems: the optimisation of the trajectory of the UAVs and the offloading optimisation. Since the vehicle locations were continuous, the module developed to optimise the trajectories was based on DDPG, while the

Table 6
Summary of literature related to the specific use cases (LAT: Minimise Application Latency; EN: Minimise Energy Consumption; QoS: Maximise user QoS). Regarding App. partitioning: ∼, allowed, but data dependencies ignored; √, allowed and data dependencies considered.

Article	Algorithm	Objective	Approach			App.		Net. Architecture				Time-varying Aspects					
			Centralised	Distributed (End Devices)	Distributed (Edge Servers)	Multi-Application	Application Partitioning	Multi-Device	Multiple BSs	Multiple Edge Servers	Cloud	Communication	Energy	Task Arrival	Device Movement	Server Location	Server Cache
Shahidinejad et al. (2021)	QL	LAT, EN		✓		✓		✓	✓	✓	✓	✓		✓			
Du et al. (2020)	A3C	EN, QoS	✓			✓						✓					
Lin et al. (2021)	AC	EN			✓	✓		✓	✓	✓		✓		✓			
Xiao et al. (2021)	AC	LAT, EN	✓			✓		✓	✓	✓	✓	✓		✓	✓		
Yuan et al. (2020)	A3C	LAT, EN, QoS	✓			✓			✓	✓	✓			✓	✓		
Wang and Guo (2021)	DQN	LAT, EN	✓			✓		✓				✓		✓	✓		

module to optimise the offloading was based on DQN, as the offloading variables were discrete.

In turn, Zhang et al. (2021e) studied UAV-assisted MEC networks with multiple users and a single UAV which carried a MEC server. In this scenario, authors proposed a DQN-based algorithm to maximise the number of tasks and minimise the energy consumption of the whole system. This algorithm controlled the proportion of offloaded tasks and the UAV trajectory. Wang et al. (2020c) also proposed a computation offloading scheme for an UAV-assisted MEC system. In this environment, wireless devices could offload their tasks to the MEC server for execution, with the addition that they could use UAVs as MEC servers with lower power to offload computation tasks, leveraging their variable location. In this UAV-assisted system, the authors proposed a DDQN-based algorithm for computation offloading. The authors continued this research line in Ke et al. (2021a), proposing in this case a distributed DQN-based algorithm, where each end device hosted an agent.

There are other articles in this case study that employ gradient-based RL algorithms, such as the A2C algorithm proposed by Zhu et al. (2020), who considered a UAV-enable MEC scenario that provided the shortest response time. For this, the proposed scenario included a cluster of UAVs acting as edge servers, executing the applications from different devices assigned by a BS. Another example of articles that use algorithms based on policy gradient is Wang et al. (2021), where the authors studied the problem of computation offloading in a UAV-assisted MEC system. In this system, a single moving UAV provided execution resources to multiple user devices. To minimise application latency of these devices, the authors optimised user scheduling, task offloading ratio, UAV flight angle and flight speed using a DDPG-based algorithm running on the UAV. A very similar perspective can be found in Wei et al. (2021), who considered the use of a network of UAVs that worked as edge servers enabling mobile devices to offload their tasks. However, in this scenario, the authors proposed a distributed algorithm based on DDPG, with cooperative exploring and prioritised experience replay. This algorithm allowed devices to cooperatively learn new policies, take into account the movement of UAVs and even possible UAV failures. Another DDPG-based approach in a UAV-assisted multi-device MEC system was proposed in Dai et al. (2021). In this article, the authors considered multiple ground devices that could offload their applications to multiple UAVs equipped with MEC servers directly, using direct line-of-sight transmissions or to a BS with a MEC server (indirectly, through the UAVs). Here, the authors proposed a DDPG-based algorithm, which was executed on the MEC server associated to the BS, to minimise the energy consumption of end devices and UAVs, while satisfying the delay requirements of the applications. Solutions based on DDPG algorithm were also proposed in Seid et al. (2021b,a), where authors studied the problem of computation offloading and resource allocation in a dynamic aerial to ground network. This network included a macro BS with a MEC server, multiple small BS with MEC servers, multiple UAVs and multiple IoT devices. In this network, UAVs formed clusters that enabled IoT devices to offload their tasks for execution, especially in cases of local server saturation or natural disasters. In order to find the optimal solution to the computation offloading and resource allocation problem, the authors proposed two Multi-Agent DDPG-based algorithms. Both algorithms were trained centrally and executed in a distributed manner. In Seid et al. (2021b), the actor network was executed by the IoT devices, while in Seid et al. (2021a) the actor network was executed by the heads of the UAV clusters. Li et al. (2021b) studied a scenario where multiple end users could offload their tasks to multiple MEC servers via a mobile UAV that could also execute the offloaded applications. In this case, the authors aimed to minimise latency, energy consumption and the size of the dropped applications, for which they proposed the use of a Soft Actor Critic (SAC) algorithm. This algorithm represents a bridge between stochastic policy optimisation and DDPG approaches, optimising a stochastic policy in an off-policy manner.

Finally, Mohammed et al. (2020) proposed an architecture to maintain the QoS of a MEC network with IoT devices in the case of a natural or human-made disaster. For this purpose, they proposed to use UAVs acting as BSs to recover the network. In addition, to secure the system against possible vulnerabilities, they proposed the use of blockchain in their multi-UAV-assisted MEC architecture. However, in this article the authors only proposed the architecture and the use of a RL-based algorithm, and did not provide simulations or real implementations.

The solutions proposed by the authors considering the use of UAVs are very varied, with the use of QL and DDPG standing out slightly, as show in Table 5. In the same way, the objectives most commonly considered by the authors are to minimise application latency and energy consumption, similar to other types of networks. Regarding the algorithm approach, the most commonly used is the centralised approach. However, the ratio of the number of articles considering a distributed approach in edge servers is higher than in other types of networks, due to the use of UAVs, which implement distributed agents, as edge servers. This also means that networks with multiple edge servers are also considered in almost all articles, as well as the location of these servers is considered to be time-varying.

5.4. Specific use cases

Although the most common case studies are included in other subsections, some articles present a particular use case, so it has been decided to include this subsection with these articles. Table 6 classifies these articles under the different criteria considered in this survey.

One of these articles is Shahidinejad et al. (2021), where the authors investigated the problem of context-aware computation offloading in a multi-user environment. Here, the authors considered the case study

of Virtual Reality (VR) Game, whose application was divided into modules, some of which could be offloaded to edge servers. In this scenario, the authors proposed a heuristic algorithm called MUCAO and a federated learning algorithm based on QL called FLUCO. This DRL algorithm was run on each mobile device, allowing collaboration between multiple devices to facilitate learning.

Another example of a VR application is Du et al. (2020), where the authors studied the use of MEC technology for VR. Due to the requirements of these applications, and in order to reduce power consumption, the authors propose a MEC system with bandwidth-rich terahertz communications to support the ultrahigh-speed wireless data transmissions required by VR. In this system, they proposed the use of an A3C algorithm, taking into account the varying conditions of the wireless channels. Although the authors described the communication channels exhaustively, they considered a scenario with a simple network architecture, with a single BS where the edge computing units were located. Lin et al. (2021) also considered a VR application. In this case, authors studied the problem of secure task offloading and resource allocation in a Wireless Virtual Reality-enabled Medical Treatment. For that, authors integrated blockchain into the system to achieve the consensus of the information of task offloading and data processing in order to resist malicious attacks. In this scenario, authors proposed a collective algorithm based on quantum-inspired AC. This algorithm was executed by each edge node, sharing their learning to adjust the reward function to avoid local optimum solution. Xiao et al. (2021) studied the offloading computation problem in a satellite-maritime MEC system that included the end-user layer, with ships and buoy sensors, the edge layer, with edge servers and base stations located on ships, the satellite link layer and the cloud server layer. In this system, the authors proposed an algorithm for computation offloading decisions. This algorithm also used an AC algorithm to characterise a dynamic threshold for the urgency of offloading tasks, performing task offloading based on the value of this threshold. Yuan et al. (2020) focused on the migration and offloading problem in the field of Software Defined Wireless Body Area Networks for smart health monitoring. For that, they also proposed an A3C-based algorithm based, which optimises the neural network with asynchronous gradient descent.

Wang and Guo (2021) proposed an edge computing solution for swarm robotics using a mobile server to offload their tasks. Authors proposed a DQN-based algorithm to reduce the energy consumption and computation latency.

As can be seen in Table 6, the number of articles proposing a specific applications like VR or robotics in the context of computation offloading in edge computing systems is currently very limited, but this trend will very probably change in the near future.

5.5. Generic studies

This subsection covers all those articles with a generic system with no specific study cases, being the end devices denominated mobile devices, user equipment or mobile users. The classification of these articles in the different points is shown in Tables 7 and 8.

This subsection starts with articles that use RL-based solutions, and then focuses on DRL-based solutions. Although most of the articles clearly describe which method they used or based their proposed solution on, some of them only indicate that their solution is based on RL. One of these articles is Zhang et al. (2020g), where the authors investigated the computation offloading problem in an environment that considered heterogeneous computational resources, channel state, task type and input data size. In this environment, they proposed a greedy algorithm to deal with the computation offloading problem, and, given the large data size, the use of an RL-based algorithm that enabled communication overhead to be reduced. Yang et al. (2020b) describe the method proposed, although it is an unusual algorithm in the articles reviewed: an Adversary Multi Armed Bandit (MAB) algorithm, which they extended with delayed feedback. Using this technique, the authors investigated the task peer offloading problem in an edge network with the objective of minimising latency. The authors also proposed the use of the MAB algorithm in Yang et al. (2021), in this case executed on the devices of the users, with the load conditions of the servers being unknown to them.

Focusing on MDP methods, we find on the one hand articles presenting gradient-based solutions, such as the solution proposed by Qiu et al. (2021), who proposed a distributed and collective AC-based DRL algorithm to deal with the problem of computation offloading in a multi-user MEC system. This algorithm was trained with experiences and knowledge from multiple environments in order to give it a more practical perspective and improve the performance of the different agents. This article is particularly notable for its experimentation in real-world scenarios. Bi et al. (2021b,a) studied the computation offloading problem in a multi-user network where they considered stochastic transmissions and task arrivals. Here, the authors proposed a framework called LyDROO, which combined Lyapunov optimisation with DRL, in particular an AC-based algorithm, to optimise the computation rate performance. Sun et al. (2021a) also proposed an AC-based algorithm for computation offloading decision making. The main difference with other papers is that learning could be performed directly from a graph representing the network topology, which was first processed by a Graph Neural Network (GNN). This allowed to adapt it to different network topologies, as well as to consider the mobility of user devices. In this proposal, both the state of the MEC system and the state of the tasks are constructed as acyclic graphs. Liu et al. (2021a) also proposed an AC-based solution, in this case a federated algorithm, with one agent in each MEC server. These authors considered a MEC system with multiple end devices and multiple MEC servers, each connected to a BS, where the federated AC algorithm was responsible for making offloading decisions, as well as radio and MEC computing resource allocation. Another article proposing an AC-based solution is Feng et al. (2020), where the authors focused on the problem of security and privacy. They proposed the use of a cooperative computation offloading and resource allocation framework for blockchain-enabled MEC systems, ensuring the reliability and irreversibility of data in MEC systems, as well as the use of an A3C algorithm, in order to maximise the computation rate and transaction throughput of blockchain systems. The main limitation of this article can be found in the consideration of a scenario lacking in dynamism, as the authors only considered communications as time-varying.

On the other hand, we find value-based methods, which can be further divided into on-policy and off-policy methods. Within the first group, we find solutions such as the one of Alfakih et al. (2020), who proposed a Sarsa algorithm to solve the computation offloading problem in a multi-server, multi-user MEC system, where different edge networks were established with their own edge servers. This provided more offloading options for user devices, allowing tasks to be executed locally, on the nearest edge server, on the adjacent edge server, or on the remote cloud. Although the network architecture considered by the authors was complex, with multiple devices, BSs and servers, the scenario lacked dynamism, with no time-varying characteristics.

Finally, among the off-policy algorithms, there are solutions based on QL. Numerous examples of papers proposing solutions based on this algorithm can be found, such as Jiang et al. (2020c), Zhou et al. (2021), Nduwayezu et al. (2020), Liang et al. (2020b), Ge et al. (2020), Mao et al. (2020), Gao et al. (2020), Yang et al. (2020a), Kiran et al. (2020), Wang et al. (2020g), Ho and Nguyen (2020).

Jiang et al. (2020c) investigated a task execution scheme for offloading decision and resource allocation to minimise energy consumption. For this purpose, authors modelled the problem as a Mixed Integer Non-Linear Programming (MINLP) problem, and then proposed a QL-based algorithm to obtain the optimal policy. The authors extended their work in Zhou et al. (2021), where they jointly considered the problems of the offloading decision, spectrum resource allocation, and

Table 7
Summary of literature related to the Generic Case Study (LAT: Minimise Application Latency; EN: Minimise Energy Consumption; CR: Maximise Computation Rate; SEC: Maximise System Security; COST: Minimise Economic Cost; CT: Maximise Completed Tasks; ER: Minimise the use of Edge Resources (computation, communication and/or storage); REP: Maximise Server Reputation). Regarding App. partitioning: ∼, allowed, but data dependencies ignored; √, allowed and data dependencies considered.

Article	Algorithm	Objective	Approach			App.		Net. Architecture				Time-varying Aspects					
			Centralised	Distributed (End Devices)	Distributed (Edge Servers)	Multi-Application	Application Partitioning	Multi-Device	Multiple BSs	Multiple Edge Servers	Cloud	Communication	Energy	Task Arrival	Device Movement	Server Location	Server Cache
Zhang et al. (2020g)	RL-based	CR	✓			✓		✓	✓	✓		✓		✓			
Yang et al. (2020b)	MAB	LAT			✓	✓		✓	✓	✓		✓		✓			
Yang et al. (2021)	MAB	LAT		✓		✓		✓	✓	✓		✓		✓			
Qiu et al. (2021)	PG, AC	LAT, EN	✓			✓	∼	✓	✓			✓		✓			
Bi et al. (2021b)	AC	CR	✓			✓		✓				✓	✓	✓			
Bi et al. (2021a)	AC	CR	✓			✓		✓				✓	✓	✓			
Sun et al. (2021a)	AC	LAT	✓			✓		✓	✓	✓				✓	✓		
Liu et al. (2021a)	AC	LAT, EN, CT			✓	✓	✓	✓	✓	✓		✓		✓			
Feng et al. (2020)	A3C	SEC, CR	✓			✓		✓	✓	✓		✓					
Alfakih et al. (2020)	SARSA	LAT, EN	✓			✓		✓	✓	✓	✓						
Jiang et al. (2020c)	QL	EN	✓			✓								✓			
Zhou et al. (2021)	QL, DDQN	EN	✓			✓		✓				✓		✓			
Nduwayezu et al. (2020)	DQL	CR	✓					✓				✓					
Liang et al. (2020b)	QL, DQN	LAT, EN	✓					✓				✓					
Ge et al. (2020)	QL	LAT, EN, SEC		✓		✓	∼	✓		✓		✓					
Mao et al. (2020)	QL	EN	✓			✓		✓	✓	✓		✓	✓	✓			
Gao et al. (2020)	QL	LAT, EN	✓			✓	✓		✓	✓	✓		✓				
Yang et al. (2020a)	QL	EN	✓			✓	✓		✓	✓	✓				✓		
Kiran et al. (2020)	QL	LAT, EN	✓			✓	∼	✓	✓	✓	✓						
Wang et al. (2020g)	QL	LAT, EN	✓					✓	✓	✓					✓		
Ho and Nguyen (2020)	QL, DQN	LAT	✓			✓	∼	✓	✓	✓	✓	✓	✓	✓			
Jiang et al. (2020b)	DRL-based	EN	✓					✓	✓	✓							
Wang and Zhu (2020)	DRL-based	EN, COST	✓				✓										
Jiang et al. (2020a)	DQN	LAT, EN		✓		✓		✓				✓	✓				
Chen et al. (2021c)	DQN	LAT, EN		✓		✓		✓	✓			✓		✓	✓		
Tefera et al. (2020)	DQN	LAT, EN		✓				✓	✓			✓					✓
Tefera et al. (2021)	DQN	LAT, EN		✓				✓	✓			✓					✓
Li et al. (2020c)	DQN	LAT		✓				✓	✓	✓		✓					
Jeong et al. (2021)	DQN	LAT		✓		✓						✓					
Elgendy et al. (2021b)	QL, DQN	LAT, EN	✓			✓		✓									✓
Hao et al. (2020)	DQN	LAT	✓			✓		✓			✓			✓			
Liu et al. (2021b)	DQN	CR	✓			✓		✓				✓	✓	✓			
Tuong et al. (2020)	DQN	LAT	✓				∼	✓				✓					
Wu et al. (2020)	DQN	CT	✓			✓		✓				✓		✓			
Tuong et al. (2021)	AC, DQN	LAT, EN	✓				∼	✓				✓					
Tong et al. (2020)	DQN	LAT, EN	✓			✓		✓	✓	✓				✓	✓		
Ale et al. (2021b)	DQN	EN, CT	✓			✓		✓	✓	✓				✓			
Yu et al. (2021)	DQN	LAT, ER	✓				✓	✓	✓	✓	✓	✓					✓
Li et al. (2020a)	DQN	EN	✓			✓		✓						✓	✓		

computation resource allocation. Here, the authors first proposed a QL-based algorithm and then a DDQN based algorithm. Nduwayezu et al. (2020) focused not only on the computation offloading problem, but also and subcarrier allocation problem in a MEC Multi-Carrier NOMA system, in which they aimed to optimise the computation rate. In this scenario, the authors proposed their DRL for Online Computation Offloading algorithm, which was composed of two elements: an offline DNN and an online QL. In the offline phase, a DNN was constructed, which inferred the value function of each state–action pair to be used for the QL algorithm in the online phase. Another example is Liang et al. (2020b), who modelled the computation offloading problem as a multi-user game and analysed the existence of nash equilibrium in this scenario. To address this problem, they proposed an algorithm based on nash equilibrium and RL (Nash-QL), adding neural networks to avoid the dimensional problem. Ge et al. (2020) also proposed an algorithm based on QL to optimally choose the server in a multi-user and multi-server MEC system. The peculiarity of this article is that the authors studied the behaviour of one of the devices against different types of attacks.

Other articles propose the use of this algorithm in scenarios with a network divided into several levels, as is the case of Mao et al. (2020), who considered a MEC system in a Heterogeneous Cellular Network with a Small BS and a Macro BS, both with a MEC server. In this scenario, the authors proposed two QL-based algorithms: the first one following the traditional method and the second one applying the hotbooting technique, which enabled the action-value function to be initiated based on training data from similar scenarios. Another example of this type of scenario was used by Gao et al. (2020), who modelled an offloading and resource allocation decision process in a collaborative environment with a mobile device, a edge server and a cloud server. In this environment, the authors considered a single-chain application that could be split and executed sequentially, representing it as a DAG graph. In this scenario, the authors proposed a QL-based offloading scheme to solve the proposed problem by dividing it into two subproblems: the optimisation of the transmission power and computation frequency of the edge cloud and the optimisation of the offloading decisions. Yang et al. (2020a) also studied the computation offloading problem in a scenario with edge and cloud servers. In this

scenario, they considered a device that moved exploiting opportunistic communication to offload its tasks, proposing an algorithm based on QL to select the optimal node for offloading. Kiran et al. (2020) also considered a MEC system with edge and cloud servers, in this case with SDN. Here, the authors proposed two RL based approaches (QL and cooperative QL) to solve the computation offloading problem.

In addition to the computation offloading problem, which is usually accompanied by the resource allocation problem, other articles have taken into account the migration and handover of tasks. Wang et al. (2020g) dealt with the problem of task offloading and migration in mobility-aware MEC networks, since the mobility of the devices may cause offloaded tasks to not be correctly returned. As a solution to this problem, the authors proposed a framework based on QL. Ho and Nguyen (2020) also studied the handover problem, in addition to the problems of server selection and cooperative offloading, in a multi-server and multi-user MEC system with a dynamic environment. In this system, tasks offloaded to a MEC server could be sent totally or partially to other servers for execution in case the first server did not have enough resources to execute it. In this scenario, the authors first proposed an algorithm based on QL and then another based on DQN.

As in the previous article, the next logical step is the use of DRL with neural networks, which allows dealing with problems of great dimensions. Thus, Jiang et al. (2020b) formulated the offloading computation problem in a multi-server, multi-user MEC system as a multi-dimensional knapsack problem with constraints. To solve this problem, the authors proposed a neural network-based architecture called Multi-Pointer networks (Mptr-Net). Wang and Zhu (2020) also used a DRL-based algorithm with DNN and KNN. The authors proposed a dynamic offloading strategy for a multi-user MEC cellular network. The authors formulated the problem as an energy consumption and offloading cost optimisation problem, transforming it into a multi-label classification problem.

Other proposals also include decentralised approaches using DQNs, such as Jiang et al. (2020a), Chen et al. (2021c), Tefera et al. (2020, 2021). Jiang et al. (2020a) proposed a distributed offloading computation algorithm in a MEC system with one single BS. According to the approach of the authors, each user device decides, in a distributed manner, whether to execute the task locally or offload it to the edge, based only on its local parameters and a few information broadcasted by the BS. The authors proposed the use of the DQN algorithm in order to avoid the risk of transmission failure due to a large number of devices offloading their tasks simultaneously. Another paper where the authors proposed a distributed approach is Chen et al. (2021c), where the authors analysed the dynamic environment and communication sharing in a MEC environment. Then, the authors proposed a distributed framework in which agents running on terminals used the DQN algorithm to learn autonomously. Meanwhile, Tefera et al. (2020, 2021) investigated a framework that enabled resource-constrained devices to offload their tasks to a MEC system in a congestion-aware, adaptive and decentralised approach. To this end, they proposed a decentralised algorithm based on non-cooperative game theory and DQN in order to ensure its convergence to the optimal solution. Li et al. (2020c) also used a distributed approach. In this case, the authors addressed the problem of computation offloading in a multi-user, multi-server NOMA-MEC system, with cooperative caching of multiple servers. Here, the authors first used a Gated Recurrent Unit (GRU) algorithm to predict the popularity of tasks and cache them on the corresponding servers. Once this was done, they proposed the use of a multi-agent DQN algorithm to take offloading and caching decisions based on the results of the popularity prediction, being each user device an agent. Jeong et al. (2021) also developed a DQN-based algorithm to make task offloading decisions in order to reduce application delay which was executed in the user device. However, in this article, authors considered a single end device and single MEC server scenario. The proposed algorithm took the channel state and the task parameter to learn the best offloading strategy.

As in the previous article Elgendy et al. (2021b) also considered the problem of computation offloading and task caching, but in this case in a multi-user and multi-task MEC system with a single server. Here, the authors formulated a computation offloading and task caching model, proposing QL and DQN-based algorithms in order to provide an efficient near-optimum solution to minimise the delay and energy sum. Although the authors considered the server cache in their scenario, which is not usually addressed in other articles, no other time-varying aspects were considered in this paper. Hao et al. (2020) also studied the problem of computation offloading in an edge-cloud system with a single BS and multiple users. The authors proposed a DQN-based algorithm called Multi-Update Reinforcement Learning, which uses two multi-layer neural networks to reduce the overall latency of the applications. Other articles using DQNs with a similar network topology are Liu et al. (2021b), Tuong et al. (2020), Wu et al. (2020). Liu et al. (2021b) proposed a DQN-based algorithm to maximise the computation rate in each time frame. This article is interesting because the authors took into account the consumption and recharging of energy of the user equipment, which received the power wirelessly at the beginning of each time period. Tuong et al. (2020) used DQNs to minimise delay in the execution of device applications in a network with time-varying channels in a MEC system with NOMA. Wu et al. (2020) proposed a DQN-based algorithm in order to maximise the number of tasks completed before their maximum delay timeout. In this scenario, they took into account the variability of local and MEC server computational resources and communication channels. Tuong et al. (2021) focused on the problem of partial computation offloading and channel resource allocation in a NOMA-assisted MEC system also consisting of a MEC server with a BS and multiple user devices. The authors considered this environment with time-varying channels, and proposed an algorithm based on AC and DQN methods to find an optimal offloading policy and an optimal channel resource allocation.

In Tong et al. (2020), Ale et al. (2021b), Yu et al. (2021), authors considered a multi-user multi-server edge computing system. Tong et al. (2020) proposed a DQN-based algorithm taking into account both the generation of new tasks and the mobility between BSs of users equipped with mobile devices. Ale et al. (2021b) proposed the use of a DQN-based algorithm to select the server for task offloading, which also recommended a maximum CPU usage for each MEC server. Yu et al. (2021) considered a ultradense edge network, in which they formulated the joint problem of application partitioning, resource allocation and service caching placement. In order to minimise the task execution time and the usage of system resources, the authors designed a two-tier approach with two DQN-based algorithms. The first one was responsible for the service caching placement, while the second one took decisions about computation offloading and resource allocation. This two-tier model was trained distributedly using federated learning to protect the data privacy of the devices, although the execution was carried out in a centralised agent.

Some articles have a special feature that distinguishes them from the rest. For example, Li et al. (2020a) investigated the reduction of power consumption in a network of mobile devices with a MEC system. The main difference with other works was the use of idle user equipment for the execution of tasks from overloaded mobile devices. The authors used a DQN algorithm to deal with this problem, as well as with the user mobility. Zhang et al. (2021c) considered a hybrid edge computing network which included MEC servers, wired devices and mobile devices. The particularity of this scenario was that the wired devices could also use their idle resources to execute tasks of mobile devices and alleviate the workload of the MEC servers. Here, the authors investigated the computation offloading and shunting problem (dividing the offloaded tasks to MECs and wired devices in a certain proportion) and proposed their DRL-based computation offloading and shunting (DCOS) algorithm. This algorithm had two phases, an offline phase where the DQN technique was used to optimise the shunting ratio and an online phase where a heuristic algorithm was used to

select where each specific task was executed. This concept was also addressed by the same authors in Zhang et al. (2021b), where they considered a similar environment, including the collaboration between MEC servers and wired devices, considering this time that wired devices could become unavailable to execute tasks. In this case, the authors proposed a single algorithm, called DRLCOS, to minimise the latency of the applications, being in charge of taking the offloading and shutting rate decisions.

Another article with an unusual perspective is Elgendy et al. (2021a), where the authors also proposed a DQN-based model for computation offloading and resource allocation that added security to the data, through a security layer using the Advanced Encryption Standard (AES) cryptographic algorithm. For this model, the authors considered a multi-user scenario with a single edge server. There are several articles related to security, although they usually propose approaches based on the use of blockchain. In one of these articles, Li et al. (2020e) focused on the security issues involved in data exchanges and resource transactions between users in an edge computing system. In the studied system, idle resources on user devices could be used by other users. In this approach, the authors proposed a blockchain-based consensus protocol in a edge environment, where the blockchain acted as a trusted third party to maintain transactions between users. In order to improve the throughput of the blockchain, the authors proposed a method based on a hierarchical DQN which considered the trust feature of the blockchain nodes and controllers and the computing resource of the blockchain system. In neither of the last two articles reviewed (Elgendy et al., 2021a; Li et al., 2020e) did the authors take into account dynamic aspects in the proposed scenario. Guo et al. (2020) developed a framework also based on blockchain for computation offloading and resource allocation in wireless networks with MEC systems. For this purpose, the authors proposed a consensus protocol in this distributed wireless network scenario. Besides maximising the throughput of the blockchain system, the objectives included minimising the latency of user tasks. In order to improve the MEC and blockchain systems, the authors proposed in this case the use of a Double-Dueling DQN algorithm to optimise different parameters, namely: the spectrum allocation, block size, and block number. This algorithm was compared with the DQN, Double DQN and Dueling DQN algorithms in order to demonstrate its faster convergence and higher accuracy in approximating the Q function. Nguyen et al. (2021) studied the computation offloading problem with access control. For that, the authors first proposed an access control mechanism enabled which used smart contracts and blockchain to improve the security, prevent malicious offloading access and preserve cloud resources. Then, they proposed an offloading scheme also based on Double-Dueling DQN algorithm to solve the joint optimisation problem of task offloading decision, edge resource allocation, bandwidth allocation and smart contract usage. Fan et al. (2021) also took into account the security in a multi-user, multi-server MEC system. In this case, the authors studied the problem of computation offloading and resource allocation decision taking in order to reduce the delay and energy consumption of the applications. In addition, they also used blockchain for the system security. In this scenario, the authors proposed an algorithm based on DDQN to take the offloading decisions, as well as those decisions related to the block size and block interval of the blockchain.

Similarly to the previous article, other authors propose the use of DDQN methods due to the difficulty of space, such as Fang et al. (2021), Zhang et al. (2020f), Liu et al. (2021c), Wang et al. (2020a), Zhang and Xu (2020), Zhang et al. (2021a), Tang and Wong (2020). Fang et al. (2021) considered a MEC environment with multiple end devices and multiple MEC servers connected to BSs. In addition, the authors modelled a task queue on the Edge servers, allowing offloaded applications to be stored for execution until the completion of previous tasks. In this scenario, the authors proposed a DDQN algorithm for computation offloading decisions to minimise application delay and system energy consumption. Zhang et al. (2020f) considered a edge computing system in an ultra-dense network with numerous BSs where a user with a smart device moved in the network through the different BSs, with dynamic task arrivals, user movement, communications and remaining device power. The authors continued their work in Liu et al. (2021c), where they proposed an algorithm based on DDQN and a context-aware attention mechanism to adaptively assign different weights to action values. This algorithm had a distributed approach, running on the user device and updating itself with the experiences generated by the environment, although it was trained centrally on a more powerful server using simulations to create initial network parameters. In turn, Wang et al. (2020a) investigated the multi-user computing offloading problem in a Cloud-Assisted Mobile Edge (CAME) computing scenario. Zhang and Xu (2020) also tested a DDQN algorithm with a distributed approach designed for computation offloading in a MEC system. In this case, the authors modelled the Stealthy Interference Attack (SIA), which causes a disturbance at the input of the DDQN algorithm. Zhang et al. (2021a), considered a single cell MEC system equipped with an Intelligent Reflecting Surface (IRS), a technology that improves the performance of wireless communication systems by adjusting the wireless propagation channel using different elements. Moreover, the devices could both receive data and harvest energy from the AP. In this scenario, the authors also proposed a DDQN algorithm with a distributed approach, in this case to take the offloading decision in order to minimise latency, energy consumption and the prize of renting the MEC system. Also from a distributed approach, Tang and Wong (2020) proposed a DRL algorithm, which included Dueling DQN and DDQN techniques, to allow mobile devices to take decisions about task offloading. For this, the authors considered a multi-user scenario where the decisions of other mobile devices could vary the conditions of the MEC system, while other devices had no information about these changes.

Other articles propose alternative versions of the DQN algorithm, as is the case of Sun et al. (2021b), who proposed a Deep Q Noisy Network (DQNN) algorithm in multi-server MEC scenario where a single device made the decision of offloading its task to one of the MEC servers. This algorithm utilised the noisy linear network to automatically adjust the level of noise for the exploration of the smart device.

DQN algorithms are sometimes used in combination with other DRL algorithms such as DDPG, like in Liang et al. (2020a), Zhan et al. (2020a). Liang et al. (2020a) designed a strategy for computation offloading and resource allocation of a partially offloaded task to the MEC server. The authors first divided the tasks into two subtasks, one of which was executed locally and the other offloaded to the edge server near the BS. To determine the best portion of the task to offload, the local execution power and the transmission power, they proposed two algorithms based on DRL: DQN and DDPG, which were executed independently at each User Equipment (UE). Zhan et al. (2020a) also proposed a decentralised computation offloading approach in an environment with different devices which had to take into account the decisions of other devices. For that, the authors designed a decentralised algorithm for computation offloading based on DDPG to achieve the optimal offloading strategy. Unlike the previous article, in this case the authors considered a slightly more complex network architecture, with multiple BSs. However, the complexity due to dynamic aspects is reduced to the field of communications.

Concerning gradient-based RL methods, we find different proposals based on the PPO algorithm, such as Li et al. (2021a), Mo et al. (2021). Li et al. (2021a) proposed a PPO-based algorithm to select the MEC server in a multi-user multi-server MEC system. The authors formulated this problem as a two-step optimisation problem, solving it with the Nash equilibrium of the strategy game. The most remarkable aspect of this article is that, unlike most articles, the authors did not take into account communications and energy consumption, but focused on obtaining the server with the highest reputation, which was given by the price of the server at a given time, its level of congestion and the percentage of previous tasks that had been successfully completed. Mo

et al. (2021) also proposed a PPO-based solution to take computation offloading and resource allocation decisions. In this case, the authors studied an edge computing system with multiple end devices, BSs and edge servers, where they considered task queues in both end devices and edge servers.

Of course, some articles propose the use of solutions only based on the DDPG algorithm, as is the case in Qinghua et al. (2020), Chen et al. (2021d), Ren and Xu (2021), Chen and Wu (2021), Chen and Wang (2020), Chen et al. (2021e), Dai et al. (2020). Qinghua et al. (2020) proposed a DDPG approach as a strategy for computational offloading and resource allocation in an environment with multiple mobile devices and a BS with a MEC server. Chen et al. (2021d) also studied the problem of computation offloading and resource allocation in a multi-user MEC system. For that, they proposed a Temporal Attentional Deterministic PG (TADPG), based on DDPG. This algorithm was executed in each mobile device to take dynamic decisions on partial task offloading and resource allocation. The TADPG agent is featured with a temporal feature extraction network (which consisted of two parts, the first of them was responsible for consistently learning the local features of input data and the second one that captured and mined the information hidden in long-distance sequential patterns) and a rank-based priority experience replay (which stored historical experiences with different importance to train the networks). Ren and Xu (2021) also considered a multi-user environment, but in this case a MEC system with energy harvesting. Here, the authors proposed a DDPG-based algorithm to take optimal offloading decisions to minimise device consumption. Chen and Wu (2021) also considered energy consumption and energy harvesting in their MEC system with mobile devices, where they investigated the joint problem of partial offloading and resource allocation. For this, they defined a scenario with a single edge server and a single BS, in which multiple devices could offload their tasks. Then, they proposed a centralised algorithm that took advantage of the graph-based relationship deduction ability from Graph Convolutional Networks (GCNs) and the self-evolution ability from the experience training of DDPG. Another example is Chen and Wang (2020), where the authors proposed a DDPG-based algorithm to offload tasks from devices in a multi-input multi-output enabled multi-user MEC system with stochastic wireless channels and task arrivals. The main difference with other algorithms is its decentralised approach, where the offloading decision is taken by each user device according to its local observation. Chen et al. (2021e) considered a NOMA-based multi-user MEC system with a single multi-antenna BS and a single edge server where they also proposed two different decentralised-approach solutions. Thus, the authors first proposed a multi-agent DDPG algorithm, which used fully observable critics and locally executable actors (strategy of centralised training and decentralised execution). In this algorithm, each user agent had a full observable critic placed in the BS and a local actor making decisions only based on its local observation. Secondly, authors proposed a Parameter Shared Multi-Agent DDPG algorithm, which trained only one actor-critic network for all user agents, attempting to learn a generic actor function for all users, in order to further reduce the computational complexity.

Dai et al. (2020) designed a DDPG algorithm to optimally solve the computation offloading and resource allocation problem in a multi-user system with multiple edge servers, including in this case a cloud server. Moreover, the authors also took into account the variability of communications over wireless channels when modelling the real world. The weakest point of this article can be found in the applications, as the authors considered only one application with no division by end device, without considering neither the task arrival as a time-varying aspect. Zhang et al. (2021d) also proposed a DDPG-based algorithm to minimise the completion time of all computation tasks. In this case, the authors studied the computation offloading problem in a multi-server MEC system consisting of one macro BS and different small BS, each of them with a edge server. In this system, the authors considered multiple users with mobility, which offloaded their tasks to edge servers (local computation is not considered). Huang et al. (2021a) also focused on the computation offloading problem in multi-server MEC systems with small cell networks, in which they considered the possible interference between devices among different cells. For this, the authors proposed a multi-agent algorithm based on DDPG, which enabled each of the BSs to select the offloading policy of each device to minimise energy consumption, taking into account the communications of nearby BSs. Chen et al. (2021b) also proposed a distributed DDPG-based algorithm to fully use the idle edge computing and storage resources in a multi-server fog system. In this distributed algorithm, the agents in the mobile devices only knew the local state, although they could communicate and collaborate with each other to offload the tasks. Moreover, the authors proposed the competition of the agents in groups, where the agents competed with their capability of dealing with a task. Another solution based on distributed DDPG is proposed by Ke et al. (2021b), who studied the problem of partial computation offloading and bandwidth allocation in a multi-user and multi-server MEC system. Furthermore, they took into account the importance of security in this system by using encryption algorithms for data communications. In this scenario, the authors proposed a distributed algorithm based on Double DQN, which was executed in each of the wireless devices to decide whether to offload and the bandwidth ratio to be used following the optimal policy.

Nath et al. (2020) also studied the use of DDPG in the computation offloading problem in a MEC system with a BS and several mobile devices. Given the limited computational resources of the MEC server, the algorithm took its decision with respect to three parameters: whether to offload a task, which wireless channels to use, and how many MEC resources to allocate. This work was extended in Nath and Wu (2020), where the authors proposed a system that took into account stochastic task arrivals and wireless channels. Furthermore, apart from the decision to offload the task for execution, the work also considers the transmission power of the devices when sending data. Other authors use algorithms that are a modification of the DDPG algorithm, as is the case of Huang et al. (2021b), who considered a multi user and multi server MEC system, with both Real Time and Non-Real time task. Here, authors formulated the problem as a Partially-Observable Markov Decision Process, which was solved using the algorithm proposed, called POTD3 (Partially-Observable Twin Delayed DDPG). Moreover, the authors proposed the use of the Dynamic Voltage and Frequency Scheduling (DVFS), which enabled mobile device to adjust CPU frequency in order to optimise the energy consumption. Another solution based on modifications of the DDPG algorithm can be found in Hu et al. (2021), where the authors proposed an algorithm, based on Twin Delayed DDPG (TD3), focused on taking both offloading and power transmission decisions in order to reduce latency and power consumption.

As can be seen in Tables 7 and 8, these articles consider multiple options in the different categories. Regarding the algorithms used, the most common are RL, with QL in the lead; DQN and DDPG. This greatly depends on the dimensionality and the type of variables of the scenario considered by the authors.

6. Discussion

This section provides an analysis and discussion of the articles that present solutions based on RL to the problem of computation offloading in MEC systems presented above. For this purpose, this section is divided into different subsections, where we will answer each of the research questions stated in Section 2.

6.1. Use cases

This subsection focuses on **RQ1**: In which use cases, related to computation offloading and MEC, are RL approaches used?

Table 8
Summary of literature related to the Generic Case Study (LAT: Minimise Application Latency; EN: Minimise Energy Consumption; CR: Maximise Computation Rate; SEC: Maximise System Security; COST: Minimise Economic Cost; CT: Maximise Completed Tasks; REP: Maximise Server Reputation; NET: Minimise the use of network resources). Regarding App. partitioning: ∼, allowed, but data dependencies ignored; √, allowed and data dependencies considered.

Article	Algorithm	Objective	Approach			App.		Net. Architecture				Time-varying Aspects					
			Centralised	Distributed (End Devices)	Distributed (Edge Servers)	Multi-Application	Application Partitioning	Multi-Device	Multiple BSs	Multiple Edge Servers	Cloud	Communication	Energy	Task Arrival	Device Movement	Server Location	Server Cache
Zhang et al. (2021c)	DQN	LAT, EN	✓			✓		✓		✓		✓		✓			
Zhang et al. (2021b)	DQN	LAT	✓			✓		✓	✓	✓		✓		✓			
Elgendy et al. (2021a)	DQN	LAT, EN	✓			✓		✓									
Li et al. (2020e)	DQN	SEC	✓			✓		✓	✓	✓							
Guo et al. (2020)	Double Dueling DQN	LAT, SEC	✓			✓		✓	✓	✓		✓		✓			
Nguyen et al. (2021)	Double Dueling DQN	LAT, EN, SEC	✓					✓			✓						
Fan et al. (2021)	DDQN	LAT, EN	✓			✓		✓	✓	✓		✓		✓			
Fang et al. (2021)	DDQN	LAT, EN	✓			✓		✓	✓	✓				✓			
Zhang et al. (2020f)	DDQN	LAT, EN		✓		✓			✓	✓		✓	✓	✓	✓		
Liu et al. (2021c)	DDQN	LAT, EN		✓		✓			✓	✓		✓	✓	✓	✓		
Wang et al. (2020a)	DQN, DDQN	LAT, EN	✓					✓			✓						
Zhang and Xu (2020)	DDQN	LAT, EN, CT		✓		✓			✓	✓				✓			
Zhang et al. (2021a)	DDQN	EN, LAT, COST		✓		✓	∼	✓				✓	✓	✓			
Tang and Wong (2020)	DDQN, Dueling DQN	LAT		✓		✓		✓	✓	✓				✓			
Sun et al. (2021b)	DQNN	LAT, EN		✓		✓	∼		✓	✓		✓		✓	✓		
Liang et al. (2020a)	DQN, DDPG	LAT, EN		✓		✓	∼	✓				✓	✓	✓			
Zhan et al. (2020a)	DDPG	LAT, EN		✓			∼	✓	✓			✓					
Li et al. (2021a)	PPO	REP		✓		✓	∼	✓		✓				✓			
Mo et al. (2021)	PPO	LAT, EN	✓			✓		✓	✓	✓		✓		✓			
Qinghua et al. (2020)	DDPG	LAT, EN	✓			✓		✓						✓			
Chen et al. (2021d)	DDPG	CT, EN		✓		✓	∼	✓						✓			
Ren and Xu (2021)	DDPG	EN	✓			✓		✓				✓	✓	✓			
Chen and Wu (2021)	DDPG	LAT, EN	✓			✓	∼	✓				✓	✓	✓			
Chen and Wang (2020)	DDPG	LAT, EN		✓		✓	∼	✓				✓		✓			
Chen et al. (2021e)	DDPG	LAT, EN	✓	✓		✓	∼	✓				✓		✓			
Dai et al. (2020)	DDPG	EN	✓					✓	✓	✓	✓	✓					
Zhang et al. (2021d)	DDPG	LAT	✓			✓	∼	✓	✓	✓				✓	✓		
Huang et al. (2021a)	DDPG	EN			✓		∼	✓	✓	✓		✓					
Chen et al. (2021b)	DDPG	LAT		✓		✓		✓	✓	✓	✓			✓	✓		
Ke et al. (2021b)	DDPG	LAT, EN, NET		✓		✓	∼	✓		✓		✓		✓			
Nath et al. (2020)	DDPG	LAT, EN	✓					✓				✓					
Nath and Wu (2020)	DDPG	LAT, EN	✓			✓		✓				✓		✓			
Huang et al. (2021b)	TD3	EN, CR	✓			✓		✓	✓	✓	✓			✓			
Hu et al. (2021)	TD3	LAT, EN, CT	✓			✓		✓				✓	✓	✓	✓		

The reviewed articles have been divided in Section 5 into the different case studies found, namely: IoT, vehicular networks, UAVs, specific use cases and generic studies or use cases.

Fig. 5 shows the proportion of each use case. Thus, we can see how the most common case study is by far the generic one, with 52.14% of the reviewed articles. In these articles the authors studied the problem of offloading computing in generic mobile networks, without describing the end application, describing the end devices as mobile devices, smart devices, user devices or just users.

Next are three case studies with a similar percentage of articles: vehicular networks (mainly characterised by the movement of vehicles, with 16.43% of the total number of articles reviewed), IoT device networks (13.57%) and UAV networks (13.57%). In this last case study, the most articles concern the use of UAVs as mobile servers (84.21% of these articles). In these articles, the authors consider these mobile servers as an auxiliary solution in case of network failure (including natural disasters) or network saturation, or to introduce the concept of computation offloading to hostile environments where traditional MEC servers are not suitable.

Finally, only in 4.29% of the articles reviewed we found proposals for specific use cases, where the authors described a particular application. VR stands out in particular, although we also found applications for smart hearth monitoring and swarm robotics.

6.2. Network and edge computing architecture

This subsection focuses on **RQ2**: Which network and edge computing architecture is considered? Are there multiple end devices and edge servers? Is a cloud layer also considered?

All of the reviewed papers consider at least one end device, one BS or AP that allows the end device to connect to the rest of the network, and one edge server that allows the end device to offload its applications. However, this network architecture is not the most interesting from the point of view of using RL algorithms to solve task placement and resource allocation in real edge scenarios. Thus, most articles extend this minimum network in one or more layers (Fig. 6). Concerning the end-device layer, 122 articles, a total of 87.14% of the reviewed articles, considered systems with multiple end-devices.

Regarding the edge layer, 84 articles, 60% of the total, considered multiple BSs or APs, while 88 articles, 62.86% of the total, considered

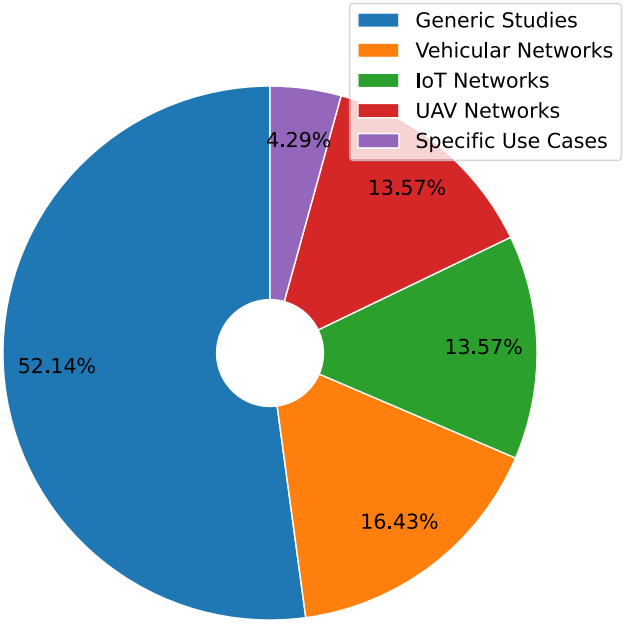

Fig. 5. The distribution of different case studies of computation offloading solutions based on RL.

multiple edge servers. However, not all articles that consider multiple BSs or APs also consider multiple edge servers or vice versa. In 6 articles, 4.23% of the total, the authors considered multiple BSs or APs but a single edge server, while in 10 articles, 7.14% of the total, they considered a single BS or AP with multiple edge servers.

In terms of the cloud computing layer, this is only considered in 35 articles, 25% of the total. In most articles, the authors who considered the cloud layer also considered multiple BSs or APs and multiple edge servers. However, in 5 articles, 3.57% of the total, the authors considered this layer together with a single BS or AP and a single edge server, while in only one article, 0.71% of the total, the authors considered multiple BSs or APs and a single edge server together with the cloud layer. Also noteworthy are another 5 articles, 3.57% of the total, where the authors proposed a network architecture that included the cloud layer, multiple edge servers and multiple BSs, but a single end device.

Finally, there are three network architectures that stand out for the number of articles in which they are considered, namely: (1) multiple end devices, with a single BS, a single edge server and without cloud layer (25.7%, in orange in Fig. 6); (2) multiple end devices, with multiple BSs and multiple edge servers but without cloud layer (29.3%, in green in Fig. 12); and (3) multiple end devices, with multiple BSs and multiple edge servers, also including cloud layer (17.1%, in yellow in Fig. 12).

Table 9 shows the distribution of the articles, in percentage, divided by the different presented case studies according to the network architecture considered by the authors. For this purpose, we have differentiated four different network layers, namely: end devices, APs/BSs, edge servers and cloud servers.

Regarding IoT networks, 94.74% of the reviewed articles related to this case study include multiple end devices. IoT networks are typically composed of a large number of low-power devices with the objective of covering a large area or performing very heterogeneous operations, which is reflected in the scenarios proposed by the authors. Furthermore, it is particularly notable that, once the end-device layer has been defined, the authors mainly consider either a simple edge network, with a single AP/BS, a single edge server and no cloud server (in 47.37% of the articles reviewed in this case study), or a complex edge network, with multiple APs/BSs and multiple edge servers, with no cloud server

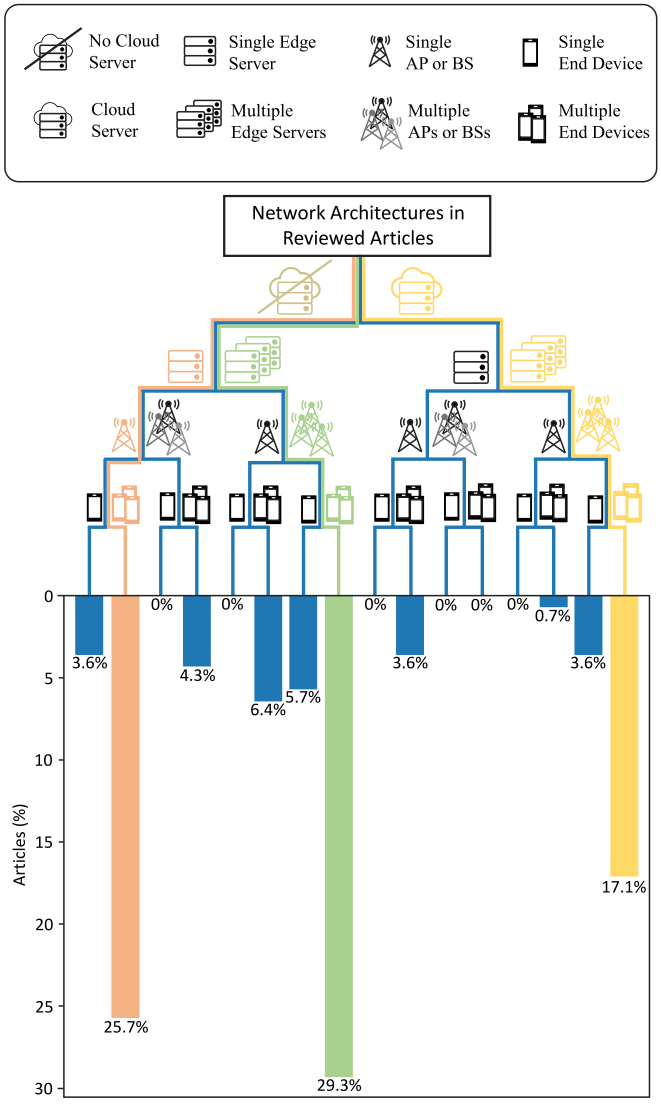

Fig. 6. Distribution of the network architectures considered in the reviewed articles.

(15.79% of the articles in this case study) or with it (26.32%). This depends greatly on the scenario proposed by the authors: for example, in a smart building, there could be a multitude of IoT devices, but all of them within the coverage range of a single AP/BS connected to a single edge server, whereas when these IoT devices are deployed over a much larger area (applied to the field of smart agriculture or smart cities, for example), a more complex network infrastructure will be required.

For vehicular networks, the authors mainly opted for network architectures closer to a possible final deployment, with multiple end devices, multiple APs/BSs and multiple edge servers, which can or not include a cloud server, representing more than 60% of the articles reviewed in this case study. In the articles of this case study, the authors usually take into account the movement of end devices, being especially interesting to analyse the handovers produced when one of these devices leaves the coverage area of an AP/BS and moves to a different one, as well as the management of offloaded applications in these cases. For this same reason, the most commonly used network architectures after those already mentioned include a network with a single end device, but with multiple APs/BSs, multiple edge servers and without or with a cloud server (13.04% and 8.7%, respectively of the case study articles reviewed).

Regarding the UAV networks case study, 47.37% of the articles reviewed in this case study propose a scenario with multiple end devices,

Table 9
Percentage of articles within each use case that consider each type of network architecture.

Network architecture				Articles (%) in each case study				
Cloud	Edge servers	APs/BSs	End devices	IoT	Vehicle	UAV	Specific	Generic
With no cloud server	Single	Single	Single	0	4.35	0	16.67	4.11
			Multiple	**47.37**	0	10.53	16.67	**32.87**
		Multiple	Single	0	0	0	0	0
			Multiple	0	4.35	0	0	6.85
	Multiple	Single	Single	0	0	0	0	0
			Multiple	0	4.35	21.05	0	5.48
		Multiple	Single	5.26	13.04	0	0	5.48
			Multiple	15.79	**30.43**	**47.37**	16.67	**28.77**
With cloud server	Single	Single	Single	0	0	0	0	0
			Multiple	5.26	4.35	0	0	4.11
		Multiple	Single	0	0	0	0	0
			Multiple	0	0	0	0	0
	Multiple	Single	Single	0	0	0	0	0
			Multiple	0	0	5.26	0	0
		Multiple	Single	0	8.70	0	16.66	2.74
			Multiple	**26.32**	**30.43**	15.79	**33.33**	9.59

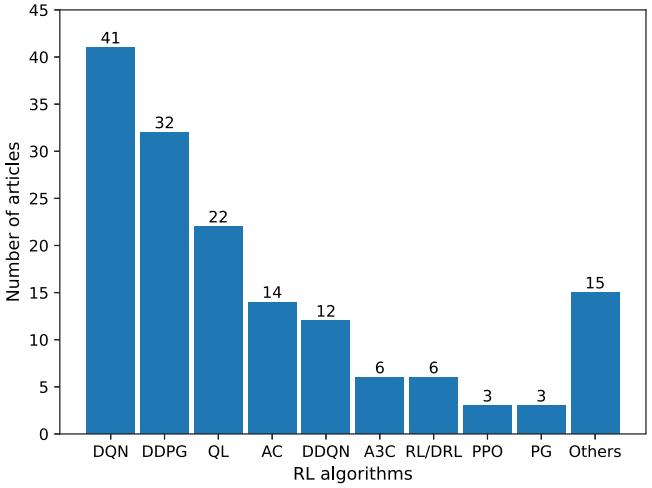

Fig. 7. Distribution of different RL techniques of the proposed solutions.

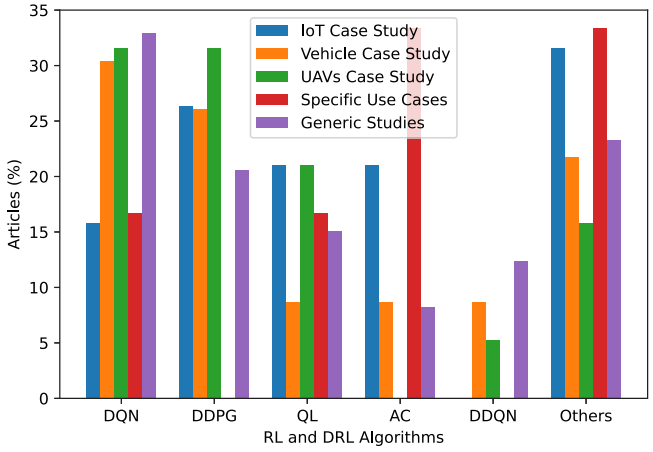

Fig. 8. Distribution of the five most widely used RL techniques (%) of the proposed solutions, divided according to the different case studies.

multiple APs/BSs and multiple edge servers, but with no cloud server. The second most proposed option (21.05% of the articles reviewed in this case study) continues along the lines of the one described above, but with a single AP/BS. These architectures (together with those of a single edge server, 10.53% of the papers in this case study) are perfectly suited to one of the scenarios where the deployment of UAVs is optimal: locations that are difficult to access, either due to location conditions or due to some kind of disaster.

With respect to the specific use cases, the authors presented different network architectures, depending on the specific application proposed in their articles. Finally, concerning the case study of generic networks, those with multiple end devices, but with a simple edge architecture (a single AP/BS and a single edge server), stand out with 32.87% of the articles reviewed included in this case study. However, the second place (28.77% of the articles reviewed in this case study) appears at the other extreme, with proposals for networks with multiple end devices, multiple APs/BSs and multiple edge servers, but without the use of cloud servers. In fact, it is particularly surprising that only a small percentage of the articles in this case study (16.44%) use cloud computing.

6.3. RL algorithms

This subsection addresses **RQ3**: Which type of RL-based algorithms are usually employed for computation offloading?

As shown in Fig. 7, DQN is on the top of the ranking with a total of 41 of the 140 reviewed articles. Closely following this result is DDPG, which is used in 32 articles. The third most used algorithm is QL with 22 articles. Following them is AC, used in 14 articles. The next most frequently used technique is DDQN, proposed as a solution in 12 of the reviewed articles. This is followed by A3C and RL/DRL-based solutions (which simply indicates that they are based on RL or DRL without referring to the specific algorithm), both with a total of 6 articles. A fewer number of articles propose the use of PPO and PG, each of them used in 3 articles. In the remaining articles we find solutions based on MAB, Dueling DQN, Double Dueling DQN and TD3, with two articles each, as well as solutions based on DQL, SARSA, DQNN, A2C, SAC, Dirichlet DDPG and Double Dueling DDPG, which only appear in 1 of the reviewed articles.

Fig. 8 shows the percentage of use of the 5 most frequently utilised algorithms in the different case studies. The case study of vehicular networks is notable, with few papers using QL and AC algorithms. This could be due to the fact that most of these papers include movement in their end devices, which increases the state space, with DRL-based algorithms working better in these cases. Something similar applies

to the case study of UAVs, where no article uses the AC algorithm, perhaps because it is not the most appropriate given the movement of the servers, which is very representative of this type of networks. The irregularity of the specific use cases is also remarkable. Although this is due to the small number of articles in this case study, there is a significant use of QL and AC algorithms. This could be due to the fact that these types of proposals tend to have a very defined network and very specific devices, making the spaces of possible actions and states to be taken more affordable.

However, one of the main problems with this point is that the authors do not usually detail the reasons why they make use of an RL algorithm or why they choose a certain algorithm to solve the computation offloading problem, beyond a few cases where they mention that traditional algorithms do not allow a real-time response to the scenario posed, or that DRL algorithms can deal with more complex scenarios. Moreover, papers generally compare one or at most two RL methods with other types of algorithms or heuristics, but lack of an exhaustive comparison between different RL algorithms in the same scenario.

6.4. Objectives

This subsection focuses on **RQ4**: What are the metrics typically optimised when solving computation offloading by means of RL techniques?

Two objectives stand out above the rest: minimising latency and minimising energy consumption (Fig. 9). Minimising latency in the execution of applications appears in 99 of the 140 articles reviewed (70.71% of the total), while minimising energy consumption, either of the end devices or of the whole system, is considered in 96 articles (68.57% of the papers).

Further below these values, we find articles proposing to maximise the number of completed task, with a total of 12 articles (8.57%). Other authors consider maximising computation rate with a total of 9 articles (6.43%) or the security of the system (closely associated with blockchain-related proposals), with a total of 8 articles, representing 5.71% of the total. Other authors propose to minimise the use of network resources, with a total of 5 articles (3.57% of the reviewed articles). Minimising the use of MEC resources and minimising economic cost are considered at least as one of the objectives of the proposed solution in 4 articles each (2.86% of the total). Maximising user QoS is considered in 3 articles, representing 2.14% of the total. Finally, the objectives of maximising load balancing, maximising the number of devices using MEC resources, maximising prediction accuracy, maximising the reputation of the selected servers and maximising transmission reliability are less common, being taken into account in only 1 article each, with 0.71% of the total.

Again, regarding the objectives or metrics used, the authors do not generally provide strong reasons for the selection of a particular set of metrics. When dealing with problems of task placement and resource allocation in an edge computing scenario, prioritising the lowest application delay or the maximum rate of processed tasks is completely reasonable. However, the use of other metrics such as maximising load balancing or maximising transmission reliability is not so obvious or straightforward, but can improve the performance of the system.

The objectives of the articles by the different case studies are presented in Table 10 and explained below. The sum of the percentages for each case study does not correspond to 100% because the same article can define multiple objectives. With regard to IoT networks, the reduction of energy consumption stands out, being considered in 78.95% of a total of 19, followed by the minimisation of application latency which appears in 73.38% of the articles. The two objectives are combined in 11 articles of 19. There are also, although in significantly fewer articles, those whose objective is to maximise completed tasks (15.79%). In all these articles, this objective is combined with either the minimisation of energy consumption or the minimisation of application latency. Finally, there is only one article (5.26%) whose sole objective is to maximise computation rate. Thus, in this case study, the authors have maintained critical objectives when offloading computation, such as reducing latency or maximising task completion, but in most cases prioritising energy reduction, an objective of vital importance in IoT networks where devices often depend on batteries or harvested energy.

As far as vehicular networks are concerned, the most popular objective is the minimisation of application latency, which appears in 69.57% of the 23 articles reviewed related to this case study. The minimisation of energy consumption is also an important objective in this case study, appearing in 52.17% of the articles. However, we also find other objectives, namely: minimising economic cost, minimising the use of MEC resources, minimise the use of network resources, maximising system security (all of them in 8.7% of the articles), maximising transmission reliability, maximising the load balancing, maximising the number of devices which uses MEC resources and maximising completed tasks (all of them in 4.35% of the articles). All these objectives focus not only on the end-device layer, but also on the edge layer and the use of network resources, as such networks tend to be more complex, taking into account the complexity of mobility and speed of end-devices.

Regarding UAV networks, again, the objectives of Minimise Application Latency (in 73.68% of the 19 articles reviewed related to this case study) and Minimise Energy Consumption (68.42%) are the main objectives set by the authors. In addition, due to the movement of UAVs, which often play the role of edge servers where devices offload their applications, network-related objectives are prioritised. Thus, in addition to the aforementioned minimisation of application latency, we find objectives such as minimising the use of network resources, maximising Completed Tasks (both of them in 10.53% of the articles), maximising the computation rate, minimising the use of MEC resources, maximising the security of the system, maximising user QoS or maximising Prediction Accuracy (all of them in 5.26% of the articles).

In the networks for specific applications, the authors especially prioritised the objective of minimising energy consumption (in 100% of a total of 6 articles), as these are applications that either run on mobile devices or propose that the edge servers have a certain degree of mobility, so energy is a point to be taken into account. In addition to this objective, the authors prioritised minimising application latency (in 66.67% of the articles) and maximising QoS (in 33.33% of the articles), in order to provide users with the optimal user experience.

In the generic devices case study, where the authors describe the end devices as mobile devices, smart devices or simply user devices, the most frequently set objectives were minimising the application latency and minimising the energy consumption, being considered in 69.83% and 68.49%, respectively, of a total of 73 articles. Other objectives are used in a more limited way, such as maximising Computation Rate (9.59% of the articles), maximising Completed Tasks (8.22%) and maximising Security (6.85% of the articles), which is linked in most of the articles to BlockChain. However, unlike other case studies, the objectives related to computation and network resources are not very common (6.85% of the articles divided into reducing the economic cost, minimising the use of MEC resources, minimising the use of network resources and maximising the reputation of the servers).

6.5. Centralised and distributed decision-making approaches

This subsection addresses **RQ5**: Are centralised or distributed approaches generally used for decision-making?

The centralised approach in which a single agent makes the computation offloading decisions for all the devices in the network stands out, being used in 88 of the 140 articles reviewed (Fig. 10). In second place we find the distributed approach where the agent is located on the end devices of the network, with 38 of the 140 articles. In 9 articles, the authors proposed a distributed approach where the different agents run on the edge servers. In Liu et al. (2020c), the authors proposed

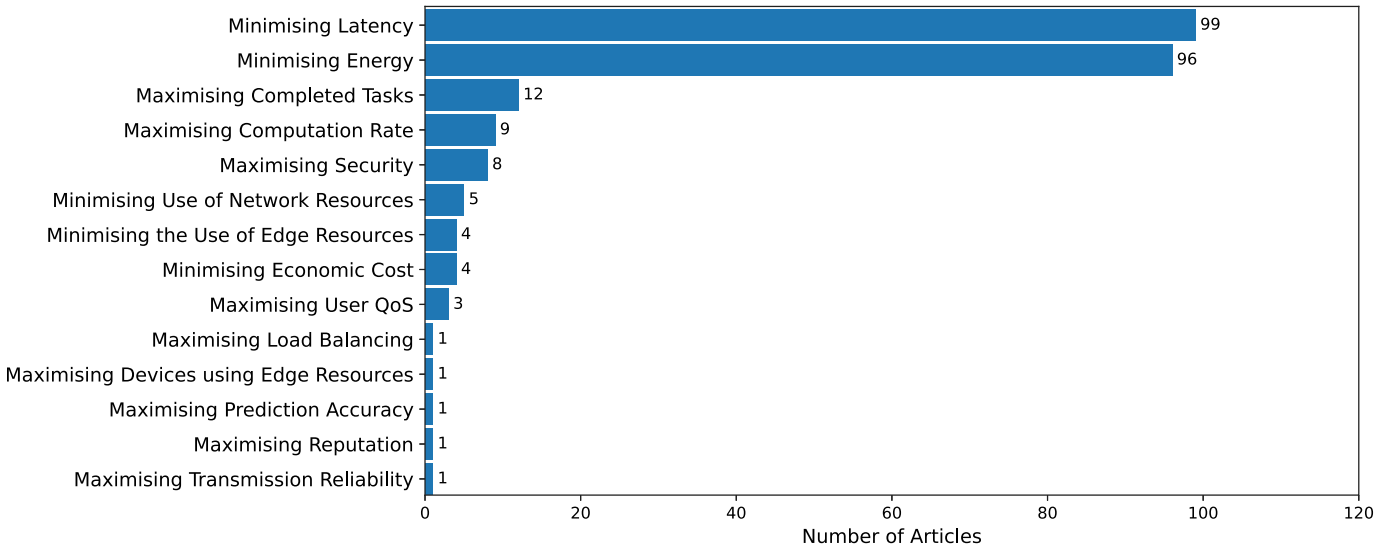

Fig. 9. Distribution of the objectives of the proposed solutions.

Table 10
Percentage of articles within each use case that consider each objective.

Objective	Articles (%) in each case study				
	IoT	Vehicular	UAV	Specific	Generic
Minimise Application Latency	**73.68**	**69.57**	**73.68**	**66.67**	**69.86**
Minimise Energy Consumption	**78.95**	**52.17**	**68.42**	**100**	**68.49**
Maximise Completed Tasks	15.79	4.35	10.53	0	8.22
Maximise Computation Rate	5.26	0	5.26	0	9.59
Maximise System Security	0	8.70	5.26	0	6.85
Minimise the use of network resources	0	8.70	10.53	0	1.37
Minimise the use of edge resources	0	8.70	5.26	0	1.37
Minimise economic cost	0	8.70	0	0	2.74
Maximise user QoS	0	0	5.26	33.33	0
Maximise the load balancing	0	4.35	0	0	0
Maximise the number of devices using edge resources	0	4.35	0	0	0
Maximise the prediction accuracy	0	0	5.26	0	0
Maximise server reputation	0	0	0	0	1.37
Maximise transmission reliability	0	4.35	0	0	0

two solutions, the first one with a centralised approach and the second one with a distributed approach on edge servers, being included in both categories. 5 articles Zhang et al. (2020c), Xu et al. (2021), Zhu et al. (2021), Seid et al. (2021b), Chen et al. (2021e) propose a distributed approach with the agents run on the end devices, but in this case the agents were trained in a centralised way. In two of these articles Zhang et al. (2020c), Chen et al. (2021e) the authors also proposed two different solutions, in this case with a centralised approach and with a distributed approach on end devices but using a centralised training, being included these articles in the corresponding categories. Only in one article Qian et al. (2021), the authors proposed a distributed approach where the agents were running on both end devices and edge servers, while only in other one Seid et al. (2021b), the authors proposed a distributed approach with the agents running on the edge servers but trained in a centralised way. Finally, only one article Mohammed et al. (2020) does not include any approach, as it is an initial concept but without actual implementation.

Fig. 11 presents the articles divided according to their case studies. As shown, the centralised approach is the predominant one in all of them, remaining between 50% and 70%, although it is slightly lower in IoT networks and UAVs case study. The articles related to these case studies propose solutions with distributed agents in a slightly higher percentage compared to the rest of the case studies.

In terms of distributed approaches, these are divided into end devices and edge servers depending on which device hosts the agent. Thus, in the IoT case study, approaches distributed on end devices are

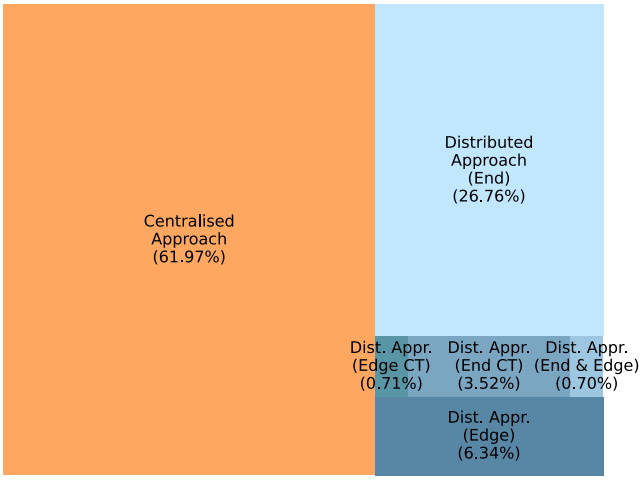

Fig. 10. Distribution of the approaches of the proposed solutions. In distributed approaches, (End) refers to agents running on end devices; (Edge) refers to agents running on edge servers; (End CT) refers to the distributed execution of agents on end devices but with a centralised training; (Edge CT) refers to the distributed execution of agents on edge servers but with a centralised training; (End & Edge) refers to the distributed approach with agents running on end devices and on edge servers.

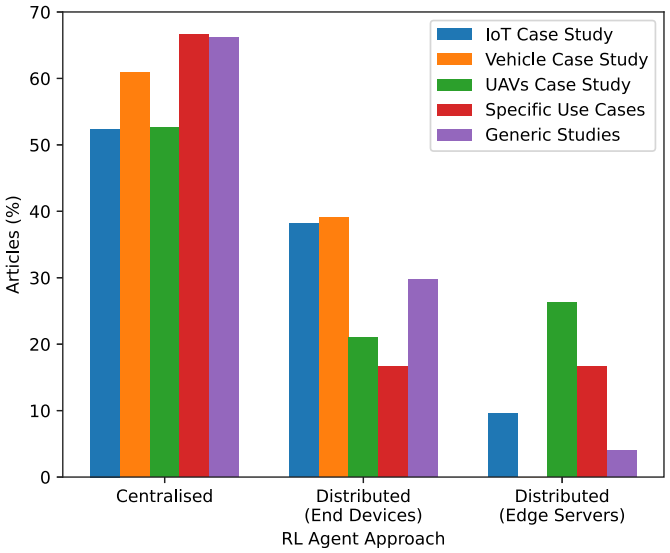

Fig. 11. Distribution of the approaches of the proposed RL techniques (%) of the proposed solutions, divided according to the different case studies.

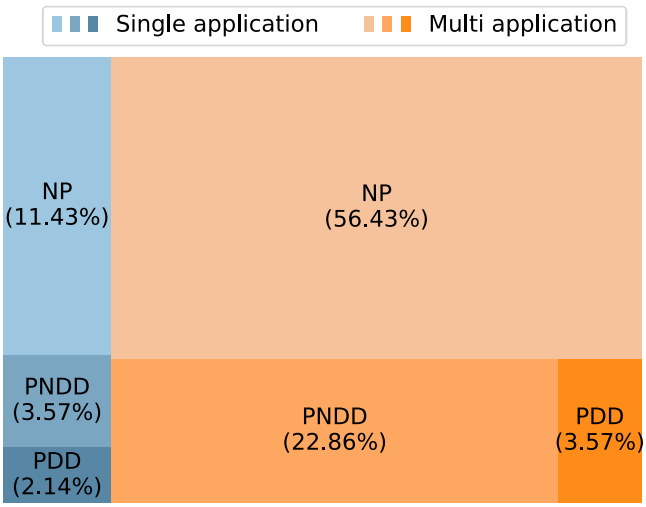

Fig. 12. Distribution of the application considerations of the proposed solutions (NP: No Partitioning; PNDD: Partitioning with No Data Dependency; PDD: Partitioning with Data Dependency).

more common than those distributed on edge servers. In IoT networks battery-powered devices that can sleep for long periods of time, without constantly communicating with the rest of the network. Therefore, an approach distributed with an agent on the end devices enables these devices to decide where to run an application.

The case study of vehicular networks has its approach distributed only on the end devices, since due to the mobility of the vehicles it is easy for them to leave the range of the agent placed in an edge server, so it is more appropriate to keep these agents in the end devices in case of a distributed approach.

This contrasts with the case study of UAV networks, where the approach is mainly distributed on edge servers. In this type of networks, UAVs are commonly used as servers in difficult access environments, to provide computing and network resources to other devices, so the most appropriate option is to host the RL agent on the UAV edge server.

The case study of specific applications has only one article in each of the distributed approaches, with the authors focusing on fulfilling the requirements of their proposed applications. In the case study of generic networks, the approach distributed on the end devices prevails with a notable difference (after the centralised approach), with some of these articles carrying out a centralised agent training and a distributed execution, allowing the authors to experiment and compare different proposals.

The choice of the approach of the RL agent is of great importance. A centralised approach is simpler to implement, and allows all network information to be considered when making decisions. However, it can become a congestion point, and fault tolerance is a critical issue. Distributed agents can increase the difficulty of implementation, especially if they work collaboratively, but they can alleviate the above problems. Moreover, they can reduce the latency in communications between the devices and the RL agent (although the impact of the communication delay between nodes and RL agents is generally omitted in the reviewed papers).

Finally, it is worthy to note that, as Tables 3 to 8 show, there is no direct relationship between the use of a centralised or a distributed approach to decision making and the type of underlying network and edge computing architecture.

6.6. Number of applications and partitioning

This subsection addresses **RQ6**: Do authors consider that end devices which offload their applications to a MEC system run a single application or multiple applications? Are these applications to be fully executed on the same platform or can they be split? If they are divided into smaller tasks, is the data dependency among the tasks of the same application taken into account?

In the articles reviewed, the authors considered end devices with multiple applications in 116 articles of the 140 reviewed articles, which represents a 82.86% of the total (orange rectangles in Fig. 12). This includes both articles where the end devices started with multiple applications and those where they started with a single application and gradually generated new applications. In contrast, 24 articles consider scenarios where end devices have a single application (in blue in Fig. 12).

Regarding partitioning, in only 45 articles, 32.14% of the total, the authors considered the option of splitting the applications into smaller tasks. Of these, 37 articles, 26.43% of the total, considered a division without data dependencies, allowing a percentage of the application to be offloaded for execution on the MEC system. In the remaining 8 articles, which means a 5.71% of the total, the authors considered splitting the applications into tasks with dependencies, representing the applications as DAGs.

Table 11 shows the distribution of the articles in the different case studies according to the scenario presented by the authors in relation to the number of applications (a single application or multiple applications on each end device) and their partitioning (no partitioning, partitioning with no data dependencies and partitioning with data dependencies between the different tasks of an application).

With respect to the IoT case study, it stands out that all articles that have a single application per device do not consider any type of partitioning. Moreover, the percentage of articles in which the authors consider multiple applications with no application partitioning or application partitioning with no data dependencies cover the most of the reviewed IoT articles. The case study of vehicular networks is more dispersed, following a similar distribution to the case study of the generic networks. Regarding the case study of UAVs, it is especially noteworthy that all articles consider multiple applications per device, although most of them do not consider any type of partitioning. Finally, in the case study of specific use cases, all the articles consider multiple applications, but with no partitioning. This may be because the authors have defined a certain number of specific applications to be executed.

The consideration of multiple applications to be allocated in the scenarios proposed by the authors is of great importance. Moreover, the partitioning of these applications, enabling them to be parallelised

Table 11
Percentage of articles within each use case according to their consideration of number of applications and type of partitioning.

Applications		Articles (%) in each case study				
Multi-App	App. Partitioning	IoT	Vehicle	UAVs	Specific	Generic
No	No partitioning	10.53	13.04	0	0	15.07
	Partitioning with no data dependencies	0	4.35	0	0	5.48
	Partitioning with data dependencies	0	4.35	0	0	2.74
Yes	No partitioning	**47.37**	**52.17**	**68.42**	**100**	**53.42**
	Partitioning with no data dependencies	**42.10**	21.74	26.32	0	19.18
	Partitioning with data dependencies	0	4.35	5.26	0	4.11

where possible, is of great interest in order to fully optimise their execution. However, few authors propose real applications detailing their division into tasks. Although at the simulation level it may be sufficient to consider standard tasks, it would be of great interest to analyse how to split real applications into different tasks to allow their execution on different devices in an edge environment and to join their outputs.

6.7. Time-varying aspects

This section focuses on **RQ7**: What characteristics of the environment are considered to be time-varying in the modelled scenarios?

There are different time-varying aspects in the considered scenarios, some of them are common to all of scenarios, while some aspects are more specific to certain use cases. For instance, the communications in a wireless environment or the generation of tasks should be reflected as time-varying aspects in most articles. In contrast, aspects such as the variation of energy with time is more relevant for the use cases of IoT or UAV networks. Similarly, the movement of end devices or servers in the case of UAV networks, or the movement of end devices in vehicular networks, are highly linked to those particular use cases. However, these aspects are not always considered by all papers associated with those use case categories (e.g., not considering the movement and handover of vehicles in vehicular networks), this being one of the main drawbacks related to the dynamic aspects considered in the reviewed articles.

The predominant time-varying characteristic in all the scenarios considered in the reviewed articles was the arrival of applications (Fig. 13). Thus, in 89 articles, 63.57% of the total, the authors considered the arrival of new applications at the end devices. The proposed algorithm had to adapt and make the decision to offload the computation of these new applications, choosing to execute them locally or to execute them on a different platform.

Secondly, wireless communications was also widely considered to be time-varying. Wireless communications are more challenging compared to wired communications, and depend on a large number of dynamic parameters in a real network. In this case, in 88 articles, 62.86% of the total, the authors considered wireless communications between end devices and BSs or APs to be time-varying.

The third most considered characteristic was the mobility of users or end devices, which was taken into account in 41 articles, 29.29% of the total. This mobility could affect the decision taking of the computation offloading both from the point of view of communications and from the point of view of the nearest edge server, as well as possible handovers or migrations of running applications. Of course, among the articles that considered the mobility of users, those related to vehicular networks stand out, where 82.61% took this mobility into account. The rest of the articles related to vehicular networks addressed the execution of applications related to vehicles, but not the mobility of vehicles.

Although not very common in traditional networks, the concept of edge computing has allowed servers to be placed closer to end devices, sometimes even in vehicles or UAVs. This has enabled certain servers to be mobile, which was considered in 20 articles, 14.29% of the total. In particular, these 20 articles can be divided between vehicular networks (20%), where servers were located in certain vehicles, and UAV networks (80%), where UAVs were used as mobile servers to provide computing resources to other devices.

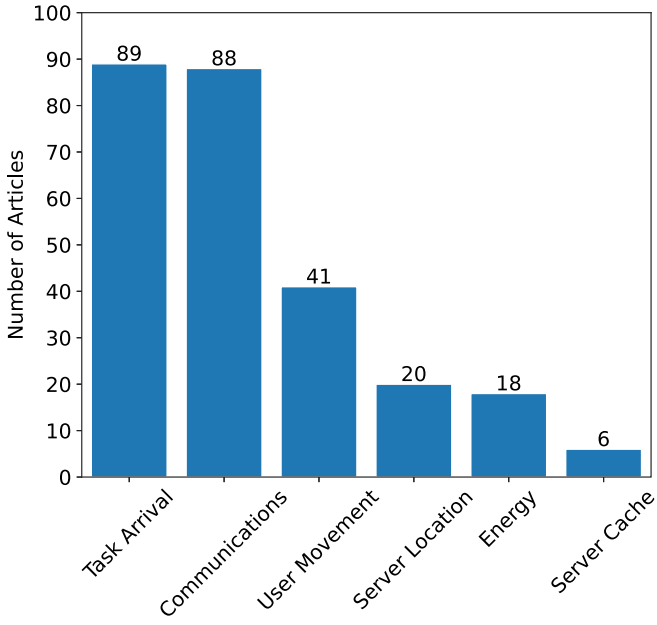

Fig. 13. Distribution of the time-varying aspects considered in the reviewed articles.

Another time-varying characteristic considered by the authors, although in a lower range, was energy. When considering this characteristic, it was not energy consumption nor energy minimisation that was taken into account as the objective of the proposed algorithms, but rather the variable energy of the devices: systems in which the devices consumed energy from their batteries but could also recharge them. In such systems, the battery charge of a device at a given time influenced the decision on computation offloading. This characteristic was considered in 18 articles, 12.86% of the total.

Finally, the last characteristic considered was the server caching. In 6 articles, 4.29% of the total, the authors considered the server cache as time-variant. In case an application was cached on a server, time and energy could be saved in communications, since the application was already on that server. This could influence the decision whether or not to offload that particular application.

Table 12 presents the distribution of the articles, in percentage, divided by the different case studies according time-varying aspects considered by the authors. In this case, the sum of the percentages is not 100%, because the time-varying aspects are not exclusive: the authors of an article can consider all of them, none of them or an intermediate number.

Regarding the IoT case study, the two time-variant aspects most considered by the authors are application arrival (63.16%) and communication (36.84%), in line with most of the other articles, although this case study is the one with the lowest percentage of articles that consider communication as a time-variant aspect. It is also noteworthy

Table 12
Percentage of articles within each use case that consider each time-varying aspect.

Time-varying aspects	Articles (%) in each case study				
	IoT	Vehicle	UAV	Spec.	Gen.
Communication	36.84	**73.91**	**68.42**	**83.33**	**63.01**
Energy	5.26	0	15.79	0.00	19.18
Application arrival	**63.16**	47.83	**84.21**	**83.33**	**61.64**
Device movement	5.26	**82.61**	36.84	50	15.07
Server location	0	17.39	**84.21**	0	0
Server cache	0	8.70	0	0	5.48

that only very few authors (5.26%) consider energy as a variant aspect in these networks, since IoT devices can harvest energy and manage it efficiently over time. This same percentage of articles (5.26%) also consider the movement of devices as time variant.

In the case study of vehicular networks is where the authors consider the movement of end-devices in by far the largest percentage of articles (82.61%). Such networks add the movement of end-devices to the challenging problem of determining which applications are offloaded. This give the possibility for an end device to leave the coverage area of one AP/BS and move into the coverage area of another, which must be taken into account when receiving the result of the offloaded application. The second most time-varying aspect considered in this case study is communication (73.91%), in many cases due to the movement of end-devices. Application arrival is also considered in almost half of the articles (47.83%). A 17.39% of these articles have considered server location as a time variable, as they propose the use of some vehicles as edge servers. Finally, this case study is where the highest percentage of articles consider server cache (8.70%).

The case study of UAV networks presents the highest percentage of articles in which the authors have considered the location of servers as a time-varying aspect (84.21%). UAVs devices can provide computing and network resources to end devices located in areas that are difficult to access, which is the scenario proposed by many authors, making it necessary to consider the movement and location of edge servers. The same percentage of articles consider application arrival. The third most frequently addressed time-varying aspect is communication (68.42%), many of them influenced by the movement of UAVs acting as APs/BSs. The next most taken into account aspect by the authors in this case study is device movement (36.84%), which in many of the proposed scenarios also involves UAVs. Also noteworthy here is the low percentage of articles that consider energy as a time-varying aspect (15.79%), given that these types of devices are usually battery-powered and energy-limited.

Concerning the specific use cases, the authors proposed communications and the arrival of applications as time-variant in most of the articles (83.33%). In 50% of the articles of this case study, the authors also addressed the movement of end devices, depending on the proposed application.

Finally, the generic networks case study shows communications as the most time-variant aspect taken into account by the authors, followed by application arrival (63.01% and 61.64%, respectively). Far behind these percentages are other aspects such as energy (even being the case study with the highest percentage), with 19.18%, device movement (more typical of other case studies), with 15.07% and server cache (being one of the two case studies that contemplate it in some articles), with 5.48%. However, no article in this case study considers server location as a time-varying aspect.

6.8. Evaluation of the proposals

This subsection focuses on **RQ8**: How did the authors evaluate and verify their proposals? Did they conduct experiments using simulations or were they conducted in testbed or real environments?

Almost all the articles reviewed validate and test their proposals through simulations. However, 79 of the articles reviewed, 56.43% of the total, indicate that they carried out simulations without describing the software tools used. In the articles where the authors specified which tools they used for the simulation, Python and its libraries were the most notable. In 34 of the articles reviewed, 24.29% of the total, the authors used Python and TensorFlow to carry out the simulations, and in one of them they also used Keras. The Keras library was also used individually in 2 articles, 1.43% of the total. PyTorch was also used in 9 of the articles reviewed, 6.43% of the total. In another 9 articles the authors referred to the use of Python in the simulations, but did not refer to the use of other libraries. On the other hand, 5 articles, 3.57% of the total, used MATLAB for the simulations performed, while 1 article used iFogsim as a simulation tool.

It is particularly striking that a single article Qiu et al. (2021), 0.71% of the total, referred to an evaluation in a real environment, in which the authors used a testbed with a laptop and several Raspberry Pis to evaluate the proposed computation offloading decision taking algorithm.

Finally, in only 4 of the 140 articles reviewed Huang et al. (2020a), Qu et al. (2021), Qiu et al. (2021), Zhan et al. (2020a), representing 2.86% of the total, the authors report on open source publishing, which we expect to be a rising trend, since it could be very valuable for experiment replication or further research.

6.9. Future directions

This subsection addresses the last research question, **RQ9**: What are the future directions of research in this field and what areas remain to be addressed?

6.9.1. Use cases

The description of the use cases often highlights the main challenges of the considered scenarios. For instance, scenarios focusing on UAV networks or vehicular networks usually address the challenge of mobile devices, and scenarios with IoT networks or UAV networks usually consider the challenge of reducing energy consumption. However, in general, the descriptions and modelling of the applications linked to the different use cases are too generic, simplified and lack realism. Therefore, further research is needed to better model the specific applications linked to each specific use case, including a characterisation of their computational load and performance requirements.

On the other hand, most papers have analysed generic scenarios, vehicular networking, UAV networks and IoT networks. However, we expect to see in the future more works related to the area of industrial applications and robotics as well as related to VR or augmented reality, topics in which some papers have already been published like Shahidinejad et al. (2021), Du et al. (2020), Lin et al. (2021), Wang and Guo (2021).

6.9.2. Real-world network architectures

Regarding network and edge computing architectures, the biggest challenge appears in the consideration of multi-device and multi-layer hierarchical networks, approximating the modelled systems to those of the real world. Furthermore, to achieve this goal, it would be desirable in future research to analyse the impact of the heterogeneity of devices, computing resources, and communication links.

6.9.3. Algorithms and comparatives

As this survey has revealed, there is a large number of very recent articles in which the authors propose solutions to the problem of computation offloading in edge computing systems based on RL. Moreover, the use of this type algorithms for computation offloading is on an increasing trend, as can be seen by comparing the articles collected by this survey with those reviewed in Shakarami et al. (2020b). Authors often propose RL solutions due to the complexity of the scenarios and the large size of the actions and states spaces, and mainly compare their proposals with other types of offloading algorithms or heuristics. However, they generally do not provide additional explanations as to

why a particular RL algorithm is selected, nor do they provide an exhaustive comparison between different RL algorithms in the same scenario, exposing the strengths and limitations of each one, which would be a desirable outcome of further research.

6.9.4. New objectives and metrics to be optimised

We have also seen in this survey that the minimisation of latency, which is a driving issue for edge computing, and energy consumption, have been extensively studied. However, in our opinion, there is a clear opportunity for research with the aim of improving the security of these systems, either through the use of blockchain (Zheng et al., 2021; Mohammed et al., 2020; Feng et al., 2020) or by considering the possibility of malicious attacks (Ge et al., 2020; Zhang and Xu, 2020), as well as adopting an economic point of view, i.e., optimising economical benefits when making decisions on computation offloading.

6.9.5. Fault tolerance in decision-making approaches

None of the reviewed articles have tested the fault tolerance of their proposals, not only against failures that prevent the agent from communicating with the rest of the devices in the network, but also from other failures that modify the network structure and require a change in the offloading policy. This is a particularly challenging point that would be worth exploring in depth.

For a centralised RL agent approach, regardless of whether it is located at the cloud layer, edge layer or at a network controller, would the system tolerate a total or temporary failure of the RL agent or its communications?

For a distributed approach, with RL agents working coordinately or not, how would a failure of one of the agents or its communications affect the system? Would it affect the whole network or would it be limited to a part of the network? In this field, more contributions are needed to address these questions.

6.9.6. Application partitioning

The articles that consider the possibility of splitting the applications into different processing tasks are approximately a third of those reviewed, and few of them considered data dependencies between tasks. Evaluating the trade-offs between performance advantages and complexity is another interesting research, also coupled with an analysis of the pragmaticity of splitting scenarios, since papers assume for instance random DAGs or DAGs created *ad-hoc* for that research, rather than extracting those graphs of dependency from real applications.

6.9.7. Exploitation of time-varying aspects

With regard to the dynamic aspects considered in the different scenarios, it can be seen that certain aspects, closely linked to particular use cases, are not taken into account in all the articles dealing with that use case. One of these aspects is user mobility, which is not taken into account in all the articles focused on vehicular and UAV networks. In fact, user mobility is an aspect that needs to be further investigated (especially in those use cases), with a focus on application migration and handovers, which have been little studied. In fact, only 5 of the articles reviewed Luo et al. (2020), Zhang et al. (2020e), Yuan et al. (2020), Wang et al. (2020g), Ho and Nguyen (2020) referred to this problem, which in our opinion deserves more attention.

Another aspect addressed in the reviewed articles, although in a limited way, is the time-varying energy of the end devices. Considering that certain devices (especially in IoT and UAV networks) could be equipped to harvest energy and increase their energy levels, the decision to offload is greatly influenced by the current energy level and the cost involved in offloading an application or executing it locally. However, despite its importance, this was a point rarely addressed in the articles reviewed, especially in articles focusing on IoT networks, with devices whose energy management is critical (being considered only in Zhang et al. (2020c)). Consequently, this aspect requires further research, particularly in those scenarios with devices that are extremely dependent on their energy level.

Similarly, the consideration of caching in servers and devices could also have a notable influence on the computation offloading decisions made by the agent, but only a few reviewed articles considered it, and additional research is required.

6.9.8. Real-world evaluations

Finally, practically all the articles reviewed verify the performance of their proposals through simulations, which, considering the relative novelty of these concepts and the difficulty of implementing them in a real environment, is understandable. However, the size of the simulated networks is often not fully representative. For example, the number of users in a smart city can be enormous, and only four articles Ale et al. (2021a), Shi et al. (2021), Zhang et al. (2021e), Ale et al. (2021b) evaluated their proposal with simulations of up to 1000 devices. Simulations with a large number of devices can help to test the scalability of the system, which is of great importance when considering MEC systems in a city area, for example. Moreover, only one of the reviewed articles carried out experiments on a real testbed. Even if they are conducted on a small scale, this may be the right way to find possible real-world challenges that are not represented in the models and simulations carried out. The development and use of testbeds to validate the proposals is one of the most important issues to be addressed in the following years. It would also be greatly useful, in order to provide a deeper understanding of the authors' proposals and experiments, if the code used would be available, where possible, as only four articles Huang et al. (2020a), Qu et al. (2021), Qiu et al. (2021), Zhan et al. (2020a) report the publication of their code.

7. Conclusion

With the increasing trend towards ubiquitous computing, devices with limited computing capabilities such as mobile devices, vehicles or IoT devices need to execute increasingly demanding applications with an appropriate delay. In this context, the use of computation offloading is of great relevance, even more in combination with the concept of edge computing, formally defined by ETSI as MEC. The edge computing paradigm enables powerful computational resources of a data centre to be available at a much closer location than in the cloud computing paradigm, greatly reducing the latency of offloaded applications. However, algorithms are needed to decide which applications should be optimally offloaded. Due to the complexity of the systems and the time-varying aspects, reinforcement learning and deep reinforcement learning algorithms are widely proposed in the literature to solve this problem.

In this survey, we have explored recently proposed RL-based solutions to this problem. These solutions were classified and compared according to multiple parameters, related both to the algorithm on which the proposed solution was based and to the system or model under consideration. This classification is summarised in Tables 3 to 12 and Figs. 5 to 13. Regarding the algorithm, in our classification we took into account the RL technique used, the objectives to be achieved by the algorithm and its approach (centralised or distributed). With regard to the model and system assumed by the authors, we took into account aspects related to the network, applications to be offloaded and time-varying aspects considered in these systems. Similarly, we also analysed the evaluation of the proposed solution, differentiating whether it was carried out through simulations or in a testbed or real environment. Finally, we discussed directions for future research, explaining the challenges and problems to be addressed in the use of RL-based algorithms for computation offloading in edge computing systems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

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

Acknowledgments

This work has been supported by Consejería de Educación de la Junta de Castilla y León and the European Regional Development Fund (Grant VA231P20) and by Ministerio de Ciencia e Innovación / Agencia Estatal de Investigación (Grant PID2020-112675RB-C42 funded by MCIN/AEI/10.13039/501100011033, Grant PID2021-124463OB-I00 funded by MCIN/AEI/10.13039/501100011033 and by ERDF A way of making Europe, and Grant RED2018-102585-T funded by MCIN/AEI/10.13039/501100011033).

References

Alameddine, H.A., Sharafeddine, S., Sebbah, S., Ayoubi, S., Assi, C., 2019. Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing. IEEE J. Sel. Areas Commun. 37 (3), 668–682. http://dx.doi.org/10.1109/JSAC.2019.2894306.

Ale, L., King, S.A., Zhang, N., Sattar, A.R., 2021a. Deep reinforcement learning aided task partitioning and computation offloading in mobile edge computing. In: 2021 IEEE/CIC International Conference on Communications in China. ICCC 2021, pp. 340–345. http://dx.doi.org/10.1109/ICCC52777.2021.9580392.

Ale, L., Zhang, N., Fang, X., Chen, X., Wu, S., Li, L., 2021b. Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning. IEEE Trans. Cogn. Commun. Netw. 7 (3), 881–892. http://dx.doi.org/10.1109/TCCN.2021.3066619.

Alfakih, T., Hassan, M.M., Gumaei, A., Savaglio, C., Fortino, G., 2020. Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA. IEEE Access 8, 54074–54084. http://dx.doi.org/10.1109/ACCESS.2020.2981434.

Alhaddadin, F., Liu, W., Gutiérrez, J.A., 2014. A user profile-aware policy-based management framework for greening the cloud. In: 2014 IEEE Fourth International Conference on Big Data and Cloud Computing. BDCloud 2014, pp. 682–687. http://dx.doi.org/10.1109/BDCloud.2014.116.

Althamary, I., Huang, C.-W., Lin, P., 2019. A survey on multi-agent reinforcement learning methods for vehicular networks. In: 2019 15th International Wireless Communications Mobile Computing Conference. IWCMC, pp. 1154–1159. http://dx.doi.org/10.1109/IWCMC.2019.8766739.

Anon, 2017. The standard, news from ETSI - Issue 2, 2017. URL https://www.etsi.org/images/files/ETSInewsletter/etsinewsletter-issue2-2017.pdf.

Beck, M., Werner, M., Feld, S., Schimper, T., 2014. Mobile edge computing: A taxonomy. In: The Sixth International Conference on Advances in Future Internet. In: AFIN 2014, pp. 48–54.

Bi, S., Huang, L., Wang, H., Zhang, Y.-J.A., 2021a. Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks. IEEE Trans. Wireless Commun. http://dx.doi.org/10.1109/TWC.2021.3085319.

Bi, S., Huang, L., Wang, H., Zhang, Y.-J.A., 2021b. Stable online computation offloading via Lyapunov-guided deep reinforcement learning. In: IEEE International Conference on Communications. pp. 1–7. http://dx.doi.org/10.1109/ICC42927.2021.9500520.

Bonomi, F., Milito, R., Zhu, J., Addepalli, S., 2012. Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing. MCC '12, Association for Computing Machinery, pp. 13–16. http://dx.doi.org/10.1145/2342509.2342513.

Cao, B., Zhang, L., Li, Y., Feng, D., Cao, W., 2019. Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework. IEEE Commun. Mag. 57 (3), 56–62. http://dx.doi.org/10.1109/MCOM.2019.1800608.

Chen, M., Hao, Y., 2018. Task offloading for mobile edge computing in software defined ultra-dense network. IEEE J. Sel. Areas Commun. 36 (3), 587–597. http://dx.doi.org/10.1109/JSAC.2018.2815360.

Chen, W., Qiu, X., Cai, T., Dai, H.-N., Zheng, Z., Zhang, Y., 2021a. Deep reinforcement learning for internet of things: A comprehensive survey. IEEE Commun. Surv. Tutor. 23 (3), 1659–1692. http://dx.doi.org/10.1109/COMST.2021.3073036.

Chen, Z., Wang, X., 2020. Decentralized computation offloading for multi-user mobile edge computing: a deep reinforcement learning approach. Eurasip J. Wirel. Commun. Netw. 2020 (1), http://dx.doi.org/10.1186/s13638-020-01801-6.

Chen, M., Wang, T., Zhang, S., Liu, A., 2021b. Deep reinforcement learning for computation offloading in mobile edge computing environment. Comput. Commun. 175, 1–12. http://dx.doi.org/10.1016/j.comcom.2021.04.028.

Chen, J., Wu, Z., 2021. Dynamic computation offloading with energy harvesting devices: A graph-based deep reinforcement learning approach. IEEE Commun. Lett. 25 (9), 2968–2972. http://dx.doi.org/10.1109/LCOMM.2021.3094842.

Chen, X., Wu, J., Cai, Y., Zhang, H., Chen, T., 2015. Energy-efficiency oriented traffic offloading in wireless networks: A brief survey and a learning approach for heterogeneous cellular networks. IEEE J. Sel. Areas Commun. 33 (4), 627–640. http://dx.doi.org/10.1109/JSAC.2015.2393496.

Chen, X., Wu, C., Liu, Z., Zhang, N., Ji, Y., 2021c. Computation offloading in beyond 5G networks: A distributed learning framework and applications. IEEE Wirel. Commun. 28 (2), 56–62. http://dx.doi.org/10.1109/MWC.001.2000296.

Chen, J., Xing, H., Xiao, Z., Xu, L., Tao, T., 2021d. A DRL agent for jointly optimizing computation offloading and resource allocation in MEC. IEEE Internet Things J. http://dx.doi.org/10.1109/JIOT.2021.3081694.

Chen, Z., Zhang, L., Pei, Y., Jiang, C., Yin, L., 2021e. NOMA-based multi-user mobile edge computation offloading via cooperative multi-agent deep reinforcement learning. IEEE Trans. Cogn. Commun. Netw. http://dx.doi.org/10.1109/TCCN.2021.3093436.

Chen, C., Zhang, Y., Wang, Z., Wan, S., Pei, Q., 2021f. Distributed computation offloading method based on deep reinforcement learning in ICV. Appl. Soft Comput. 103, http://dx.doi.org/10.1016/j.asoc.2021.107108.

Cui, Y., Du, L., Wang, H., Wu, D., Wang, R., 2021. Reinforcement learning for joint optimization of communication and computation in vehicular networks. IEEE Trans. Veh. Technol. http://dx.doi.org/10.1109/TVT.2021.3125109.

Cui, Y., Zhang, D., Zhang, T., Chen, L., Piao, M., Zhu, H., 2020. Novel method of mobile edge computation offloading based on evolutionary game strategy for IoT devices. AEU - Int. J. Electron. Commun. 118, http://dx.doi.org/10.1016/j.aeue.2020.153134.

Dai, B., Niu, J., Ren, T., Hu, Z., Atiquzzaman, M., 2021. Towards energy-efficient scheduling of UAV and base station hybrid enabled mobile edge computing. IEEE Trans. Veh. Technol. http://dx.doi.org/10.1109/TVT.2021.3129214.

Dai, Y., Zhang, K., Maharjan, S., Zhang, Y., 2020. Edge intelligence for energy-efficient computation offloading and resource allocation in 5G beyond. IEEE Trans. Veh. Technol. 69 (10), 12175–12186. http://dx.doi.org/10.1109/TVT.2020.3013990.

Davis, A., Parikh, J., Weihl, W.E., 2004. Edgecomputing: Extending enterprise applications to the edge of the internet. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters. WWW Alt. '04, Association for Computing Machinery, pp. 180–187. http://dx.doi.org/10.1145/1013367.1013397.

Deng, X., Yin, J., Guan, P., Xiong, N.N., Zhang, L., Mumtaz, S., 2021. Intelligent delay-aware partial computing task offloading for multi-user industrial internet of things through edge computing. IEEE Internet Things J. http://dx.doi.org/10.1109/JIOT.2021.3123406.

Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., Weihl, W., 2002. Globally distributed content delivery. IEEE Internet Comput. 6, 50–58. http://dx.doi.org/10.1109/MIC.2002.1036038.

Dinh, H.T., Lee, C., Niyato, D., Wang, P., 2013. A survey of mobile cloud computing: architecture, applications, and approaches. Wirel. Commun. Mob. Comput. 13 (18), 1587–1611. http://dx.doi.org/10.1002/wcm.1203.

Dong, H., Ding, Z., Zhang, S., 2020. Deep Reinforcement Learning Fundamentals, Research and Applications: Fundamentals, Research and Applications. Springer Nature Singapore Pte Ltd. http://dx.doi.org/10.1007/978-981-15-4095-0.

Du, J., Yu, F., Lu, G., Wang, J., Jiang, J., Chu, X., 2020. MEC-assisted immersive VR video streaming over Terahertz wireless networks: A deep reinforcement learning approach. IEEE Internet Things J. 7 (10), 9517–9529. http://dx.doi.org/10.1109/JIOT.2020.3003449.

Elgendy, I.A., Muthanna, A., Hammoudeh, M., Shaiba, H., Unal, D., Khayyat, M., 2021a. Advanced deep learning for resource allocation and security aware data offloading in industrial mobile edge computing. Big Data 9 (4), 265–278. http://dx.doi.org/10.1089/big.2020.0284.

Elgendy, I.A., Zhang, W.-Z., He, H., Gupta, B.B., Abd El-Latif, A.A., 2021b. Joint computation offloading and task caching for multi-user and multi-task MEC systems: reinforcement learning-based algorithms. Wirel. Netw. 27 (3), 2023–2038. http://dx.doi.org/10.1007/s11276-021-02554-w.

Elsevier, 2022b. Scopus preview. URL www.scopus.com.

Fan, W., Zhang, W., Wang, L., Liu, T., Zhang, G., 2021. Joint offloading and resource allocation in cooperative blockchain-enabled MEC system. In: ACM International Conference Proceeding Series. pp. 136–140. http://dx.doi.org/10.1145/3472634.3472666.

Fang, J., Zhang, M., Ye, Z., Shi, J., Wei, J., 2021. Smart collaborative optimizations strategy for mobile edge computing based on deep reinforcement learning. Comput. Electr. Eng. 96, http://dx.doi.org/10.1016/j.compeleceng.2021.107539.

Feng, J., Yu, F., Pei, Q., Chu, X., Du, J., Zhu, L., 2020. Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach. IEEE Internet Things J. 7 (7), 6214–6228. http://dx.doi.org/10.1109/JIOT.2019.2961707.

Ferrer, A.J., Marquès, J.M., Jorba, J., 2019. Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing. ACM Comput. Surv. 51 (6), http://dx.doi.org/10.1145/3243929.

Gao, Z., Hao, W., Han, Z., Yang, S., 2020. Q-learning-based task offloading and resources optimization for a collaborative computing system. IEEE Access 8, 149011–149024. http://dx.doi.org/10.1109/ACCESS.2020.3015993.

Ge, S., Lu, B., Gong, J., Chen, X., 2020. Computation offloading and security with Q-learning. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 316 LNICST, pp. 71–81. http://dx.doi.org/10.1007/978-3-030-44751-9_7.

Geng, L., Zhao, H., Liu, H., Wang, Y., Feng, W., Bai, L., 2021. Deep reinforcement learning-based computation offloading in vehicular networks. In: Proceedings - 2021 8th IEEE International Conference on Cyber Security and Cloud Computing and 2021 7th IEEE International Conference on Edge Computing and Scalable Cloud, CSCloud-EdgeCom 2021. pp. 200–206. http://dx.doi.org/10.1109/CSCloud-EdgeCom52276.2021.00044.

Giust, F., Verin, G., Antevski, K., Chou, J., Fang, Y., Featherstone, W., Fontes, F., Frydman, D., Li, A., Manzalini, A., et al., 2018. MEC deployments in 4G and evolution towards 5G. ETSI White Paper 24, 1–24.

Gong, Y., Wang, J., Nie, T., 2020. Deep reinforcement learning aided computation offloading and resource allocation for IoT. In: 2020 IEEE Computing, Communications and IoT Applications. ComComAp, pp. 1–6. http://dx.doi.org/10.1109/ComComAp51192.2020.9398891.

Guo, F., Yu, F., Zhang, H., Ji, H., Liu, M., Leung, V.C., 2020. Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing. IEEE Trans. Wireless Commun. 19 (3), 1689–1703. http://dx.doi.org/10.1109/TWC.2019.2956519.

Hao, H., Xu, C., Zhong, L., Muntean, G.-M., 2020. A multi-update deep reinforcement learning algorithm for edge computing service offloading. In: MM 2020 - Proceedings of the 28th ACM International Conference on Multimedia. pp. 3256–3264. http://dx.doi.org/10.1145/3394171.3413702.

Ho, T.M., Nguyen, K.-K., 2020. Joint server selection, cooperative offloading and handover in multi-access edge computing wireless network: A deep reinforcement learning approach. IEEE Trans. Mob. Comput. http://dx.doi.org/10.1109/TMC.2020.3043736.

Hu, Z., Niu, J., Ren, T., Dai, B., Li, Q., Xu, M., Das, S.K., 2021. An efficient online computation offloading approach for large-scale mobile edge computing via deep reinforcement learning. IEEE Trans. Serv. Comput. http://dx.doi.org/10.1109/TSC.2021.3116280.

Hu, Y.C., Patel, M., Sabella, D., Sprecher, N., Young, V., 2015. Mobile edge computing: A key technology towards 5G. ETSI White Paper 11, 1–16.

Huang, L., Bi, S., Zhang, Y.-J.A., 2020a. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. IEEE Trans. Mob. Comput. 19 (11), 2581–2593. http://dx.doi.org/10.1109/TMC.2019.2928811.

Huang, X., He, L., Zhang, W., 2020b. Vehicle speed aware computing task offloading and resource allocation based on multi-agent reinforcement learning in a vehicular edge computing network. In: Proceedings - 2020 IEEE 13th International Conference on Edge Computing. EDGE 2020, pp. 1–8. http://dx.doi.org/10.1109/EDGE50951.2020.00008.

Huang, X., Leng, S., Maharjan, S., Zhang, Y., 2021a. Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks. IEEE Trans. Veh. Technol. 70 (9), 9282–9293. http://dx.doi.org/10.1109/TVT.2021.3096928.

Huang, H., Ye, Q., Zhou, Y., 2021b. Deadline-aware task offloading with partially-observable deep reinforcement learning for multi-access edge computing. IEEE Trans. Netw. Sci. Eng. http://dx.doi.org/10.1109/TNSE.2021.3115054.

Islam, A., Debnath, A., Ghose, M., Chakraborty, S., 2021. A survey on task offloading in multi-access edge computing. J. Syst. Archit. 118 (102225), 1–16. http://dx.doi.org/10.1016/j.sysarc.2021.102225.

Jeong, J., Kim, I.-M., Hong, D., 2021. Deep reinforcement learning-based task offloading decision in the time varying channel. In: 2021 International Conference on Electronics, Information, and Communication. ICEIC 2021, pp. 1–4. http://dx.doi.org/10.1109/ICEIC51217.2021.9369737.

Jiang, C., Cheng, X., Gao, H., Zhou, X., Wan, J., 2019. Toward computation offloading in edge computing: A survey. IEEE Access 7, 131543–131558. http://dx.doi.org/10.1109/ACCESS.2019.2938660.

Jiang, B., Li, K., Zhou, B., Tao, M., Chen, Z., 2020a. Deep reinforcement learning for distributed computation offloading in massive-user mobile edge networks. In: The 12th International Conference on Wireless Communications and Signal Processing. WCSP 2020, pp. 811–816. http://dx.doi.org/10.1109/WCSP49889.2020.9299723.

Jiang, Q., Zhang, Y., Yan, J., 2020b. Neural combinatorial optimization for energy-efficient offloading in mobile edge computing. IEEE Access 8, 35077–35089. http://dx.doi.org/10.1109/ACCESS.2020.2974484.

Jiang, K., Zhou, H., Li, D., Liu, X., Xu, S., 2020c. A Q-learning based method for energy-efficient computation offloading in mobile edge computing. In: Proceedings - International Conference on Computer Communications and Networks. 2020-August. ICCCN, pp. 1–7. http://dx.doi.org/10.1109/ICCCN49398.2020.9209738.

Ke, H., Wang, J., Deng, L., Ge, Y., Wang, H., 2020. Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks. IEEE Trans. Veh. Technol. 69 (7), 7916–7929. http://dx.doi.org/10.1109/TVT.2020.2993849.

Ke, H., Wang, H., Sun, W., Sun, H., 2021a. Adaptive computation offloading policy for multi-access edge computing in heterogeneous wireless networks. IEEE Trans. Netw. Serv. Manag. http://dx.doi.org/10.1109/TNSM.2021.3118696.

Ke, H.C., Wang, H., Zhao, H., Sun, W.J., 2021b. Deep reinforcement learning-based computation offloading and resource allocation in security-aware mobile edge computing. Wirel. Netw. 27 (5), 3357–3373. http://dx.doi.org/10.1007/s11276-021-02643-w.

Khan, I., Tao, X., Shafiqur Rahman, G., Rehman, W.U., Salam, T., 2020. Advanced energy-efficient computation offloading using deep reinforcement learning in MTC edge computing. IEEE Access 8, 82867–82875. http://dx.doi.org/10.1109/ACCESS.2020.2991057.

Khayyat, M., Elgendy, I.A., Muthanna, A., Alshahrani, A.S., Alharbi, S., Koucheryavy, A., 2020. Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks. IEEE Access 8, 137052–137062. http://dx.doi.org/10.1109/ACCESS.2020.3011705.

Kim, K., Park, Y.M., Seon Hong, C., 2020. Machine learning based edge-assisted UAV computation offloading for data analyzing. In: International Conference on Information Networking. 2020-January. pp. 117–120. http://dx.doi.org/10.1109/ICOIN48656.2020.9016432.

Kiran, N., Pan, C., Wang, S., Yin, C., 2020. Joint resource allocation and computation offloading in mobile edge computing for SDN based wireless networks. J. Commun. Netw. 22 (1), 1–11. http://dx.doi.org/10.1109/JCN.2019.000046.

Klas, G., 2015. Fog computing and mobile edge cloud gain momentum open fog consortium, ETSI MEC and cloudlets. pp. 1–14.

Li, G., Chen, M., Wei, X., Qi, T., Zhuang, W., 2020a. Computation offloading with reinforcement learning in D2D-MEC network. In: 2020 International Wireless Communications and Mobile Computing. IWCMC 2020, pp. 69–74. http://dx.doi.org/10.1109/IWCMC48107.2020.9148285.

Li, M., Gao, J., Zhao, L., Shen, X., 2020b. Deep reinforcement learning for collaborative edge computing in vehicular networks. IEEE Trans. Cogn. Commun. Netw. 6 (4), 1122–1135. http://dx.doi.org/10.1109/TCCN.2020.3003036.

Li, S., Hu, X., Du, Y., 2021a. Deep reinforcement learning and game theory for computation offloading in dynamic edge computing markets. IEEE Access 9, 121456–121466. http://dx.doi.org/10.1109/ACCESS.2021.3109132.

Li, S., Hu, X., Du, Y., 2021b. Deep reinforcement learning for computation offloading and resource allocation in unmanned-aerial-vehicle assisted edge computing. Sensors 21 (19), http://dx.doi.org/10.3390/s21196499.

Li, S., Li, B., Zhao, W., 2020c. Joint optimization of caching and computation in multi-server NOMA-MEC system via reinforcement learning. IEEE Access 8, 112762–112771. http://dx.doi.org/10.1109/ACCESS.2020.3002895.

Li, Y., Qi, F., Wang, Z., Yu, X., Shao, S., 2020d. Distributed edge computing offloading algorithm based on deep reinforcement learning. IEEE Access 8, 85204–85215. http://dx.doi.org/10.1109/ACCESS.2020.2991773.

Li, Q., Sun, Y., Tian, T., Yang, R., Meng, L., Zhang, Y., Yu, F., 2020e. Research on security of D2D resource sharing based on blockchain in mobile edge network. In: 2020 12th International Conference on Communication Software and Networks. ICCSN 2020, pp. 202–206. http://dx.doi.org/10.1109/ICCSN49894.2020.9139065.

Li, Y., Xu, S., 2021. Collaborative optimization of edge-cloud computation offloading in internet of vehicles. In: Proceedings - International Conference on Computer Communications and Networks. 2021-July. ICCCN, pp. 1–6. http://dx.doi.org/10.1109/ICCCN52240.2021.9522252.

Li, Z., Xu, M., Nie, J., Kang, J., Chen, W., Xie, S., 2021c. NOMA-enabled cooperative computation offloading for blockchain-empowered internet of things: A learning approach. IEEE Internet Things J. 8 (4), 2364–2378. http://dx.doi.org/10.1109/JIOT.2020.3016644.

Liang, Y., He, Y., Zhong, X., 2020a. Decentralized computation offloading and resource allocation in MEC by deep reinforcement learning. In: 2020 IEEE/CIC International Conference on Communications in China. ICCC 2020, pp. 244–249. http://dx.doi.org/10.1109/ICCC49849.2020.9238942.

Liang, S., Wan, H., Qin, T., Li, J., Chen, W., 2020b. Multi-user computation offloading for mobile edge computing: A deep reinforcement learning and game theory approach. In: 2020 IEEE 20th International Conference on Communication Technology Proceedings. 2020-October. pp. 1534–1539. http://dx.doi.org/10.1109/ICCT50939.2020.9295872.

Lin, P., Song, Q., Yu, F.R., Wang, D., Guo, L., 2021. Task offloading for wireless VR-enabled medical treatment with blockchain security using collective reinforcement learning. IEEE Internet Things J. http://dx.doi.org/10.1109/JIOT.2021.3051419.

Lin, H., Zeadally, S., Chen, Z., Labiod, H., Wang, L., 2020. A survey on computation offloading modeling for edge computing. J. Netw. Comput. Appl. 169, 102781. http://dx.doi.org/10.1016/j.jnca.2020.102781.

Liu, K.-H., Hsu, Y.-H., Lin, W.-N., Liao, W., 2021a. Fine-grained offloading for multi-access edge computing with actor-critic federated learning. In: IEEE Wireless Communications and Networking Conference. 2021-March. WCNC, pp. 1–6. http://dx.doi.org/10.1109/WCNC49053.2021.9417477.

Liu, R., Liu, X., Wang, S., Yin, C., 2020a. Deep deterministic policy gradient based computation offloading in wireless-powered MEC networks. In: Deep Deterministic Policy Gradient Based Computation Offloading in Wireless-Powered MEC Networks. pp. 1–6. http://dx.doi.org/10.1109/GCWkshps50303.2020.9367589.

Liu, W., Shao, X., Wang, C., Gu, X., Jiang, F., Peng, J., 2020b. An online reinforcement learning offloading method for delay-sensitive vehicular service. In: Proceedings - 2020 IEEE 22nd International Conference on High Performance Computing and Communications, IEEE 18th International Conference on Smart City and IEEE 6th International Conference on Data Science and Systems, HPCC-SmartCity-DSS 2020. pp. 973–978. http://dx.doi.org/10.1109/HPCC-SmartCity-DSS50907.2020.00130.

Liu, W., Wang, C., Mi, J., Luan, H., Luo, Y., 2021b. A reinforcement model based prioritized replay to solve the offloading problem in edge computing. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 12939 LNCS, pp. 471–478. http://dx.doi.org/10.1007/978-3-030-86137-7_50.

Liu, Y., Xie, S., Zhang, Y., 2020c. Cooperative offloading and resource management for UAV-enabled mobile edge computing in power IoT system. IEEE Trans. Veh. Technol. 69 (10), 12229–12239. http://dx.doi.org/10.1109/TVT.2020.3016840.

Liu, X., Yu, J., Feng, Z., Gao, Y., 2020d. Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing. China Commun. 17 (9), 220–236. http://dx.doi.org/10.23919/JCC.2020.09.017.

Liu, X., Yu, J., Wang, J., Gao, Y., 2020e. Resource allocation with edge computing in IoT networks via machine learning. IEEE Internet Things J. 7 (4), 3415–3426. http://dx.doi.org/10.1109/JIOT.2020.2970110.

Liu, T., Zhang, Y., Zhu, Y., Tong, W., Yang, Y., 2021c. Online computation offloading and resource scheduling in mobile-edge computing. IEEE Internet Things J. 8 (8), 6649–6664. http://dx.doi.org/10.1109/JIOT.2021.3051427.

Liu, H., Zhao, H., Geng, L., Feng, W., 2020f. A policy gradient based offloading scheme with dependency guarantees for vehicular networks. In: 2020 IEEE Globecom Workshops, GC Wkshps 2020 - Proceedings. pp. 1–6. http://dx.doi.org/10.1109/GCWkshps50303.2020.9367544.

Liu, H., Zhao, H., Geng, L., Wang, Y., Feng, W., 2021d. A distributed dependency-aware offloading scheme for vehicular edge computing based on policy gradient. In: Proceedings - 2021 8th IEEE International Conference on Cyber Security and Cloud Computing and 2021 7th IEEE International Conference on Edge Computing and Scalable Cloud, CSCloud-EdgeCom 2021. pp. 176–181. http://dx.doi.org/10.1109/CSCloud-EdgeCom52276.2021.00040.

Long, J., Luo, Y., Zhu, X., Luo, E., Huang, M., 2020. Computation offloading through mobile vehicles in IoT-edge-cloud network. Eurasip J. Wirel. Commun. Netw. 2020 (1), http://dx.doi.org/10.1186/s13638-020-01848-5.

Lu, H., He, X., Du, M., Ruan, X., Sun, Y., Wang, K., 2020. Edge QoE: Computation offloading with deep reinforcement learning for internet of things. IEEE Internet Things J. 7 (10), 9255–9265. http://dx.doi.org/10.1109/JIOT.2020.2981557.

Luo, Q., Li, C., Luan, T.H., Shi, W., 2020. Collaborative data scheduling for vehicular edge computing via deep reinforcement learning. IEEE Internet Things J. 7 (10), 9637–9650. http://dx.doi.org/10.1109/JIOT.2020.2983660.

Luong, N.C., Hoang, D.T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., Kim, D.I., 2019. Applications of deep reinforcement learning in communications and networking: A survey. IEEE Commun. Surv. Tutor. 21 (4), 3133–3174. http://dx.doi.org/10.1109/COMST.2019.2916583.

Mach, P., Becvar, Z., 2017. Mobile edge computing: A survey on architecture and computation offloading. IEEE Commun. Surv. Tutor. 19 (3), 1628–1656. http://dx.doi.org/10.1109/COMST.2017.2682318.

Mao, M., Chai, R., Chen, Q., 2020. Energy efficient computation offloading for energy harvesting-enabled heterogeneous cellular networks (Workshop). In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 313 LNICST, pp. 391–401. http://dx.doi.org/10.1007/978-3-030-41117-6_32.

Masdari, M., Khezri, H., 2020. Efficient offloading schemes using Markovian models: a literature review. Computing 102, 1673–1716. http://dx.doi.org/10.1007/s00607-020-00812-x.

Mekrache, A., Bradai, A., Moulay, E., Dawaliby, S., 2021. Deep reinforcement learning techniques for vehicular networks: Recent advances and future trends towards 6G. Veh. Commun. 100398. http://dx.doi.org/10.1016/j.vehcom.2021.100398.

Mell, P., Grance, T., 2011. The NIST Definition of Cloud Computing, . Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, http://dx.doi.org/10.6028/NIST.SP.800-145.

Mo, R., Xu, X., Zhang, X., Qi, L., Liu, Q., 2021. Computation offloading and resource management for energy and cost trade-offs with deep reinforcement learning in mobile edge computing. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 13121 LNCS, pp. 563–577. http://dx.doi.org/10.1007/978-3-030-91431-8_35.

Mohammed, A., Nahom, H., Tewodros, A., Habtamu, Y., Hayelom, G., 2020. Deep reinforcement learning for computation offloading and resource allocation in blockchain-based multi-UAV-enabled mobile edge computing. In: 2020 17th International Computer Conference on Wavalet Active Media Techonology and Information Processing. ICCWAMTIP, pp. 295–299. http://dx.doi.org/10.1109/ICCWAMTIP51612.2020.9317445.

Mukherjee, M., Shu, L., Wang, D., 2018. Survey of fog computing: Fundamental, network applications, and research challenges. IEEE Commun. Surv. Tutor. 20 (3), 1826–1857. http://dx.doi.org/10.1109/COMST.2018.2814571.

Mustafa, E., Shuja, J., uz Zaman, S.K., Jehangiri, A.I., Din, S., Rehman, F., Mustafa, S., Maqsood, T., Khan, A.N., 2021. Joint wireless power transfer and task offloading in mobile edge computing: a survey. Cluster Comput. 101, 1–20. http://dx.doi.org/10.1007/s10586-021-03376-3.

Naha, R.K., Garg, S., Georgakopoulos, D., Jayaraman, P.P., Gao, L., Xiang, Y., Ranjan, R., 2018. Fog computing: Survey of trends, architectures, requirements, and research directions. IEEE Access 6, 47980–48009. http://dx.doi.org/10.1109/ACCESS.2018.2866491.

Nath, S., Li, Y., Wu, J., Fan, P., 2020. Multi-user multi-channel computation offloading and resource allocation for mobile edge computing. In: IEEE International Conference on Communications. 2020-June. pp. 1–6. http://dx.doi.org/10.1109/ICC40277.2020.9149124.

Nath, S., Wu, J., 2020. Dynamic computation offloading and resource allocation for multi-user mobile edge computing. In: 2020 IEEE Global Communications Conference, GLOBECOM 2020 - Proceedings. 2020-January. pp. 1–6. http://dx.doi.org/10.1109/GLOBECOM42002.2020.9348161.

Nduwayezu, M., Pham, Q.-V., Hwang, W.-J., 2020. Online computation offloading in NOMA-based multi-access edge computing: A deep reinforcement learning approach. IEEE Access 8, 99098–99109. http://dx.doi.org/10.1109/ACCESS.2020.2997925.

Nguyen, D., Pathirana, P., Ding, M., Seneviratne, A., 2021. Secure computation offloading in blockchain based IoT networks with deep reinforcement learning. IEEE Trans. Netw. Sci. Eng. http://dx.doi.org/10.1109/TNSE.2021.3106956.

Nomikos, N., Zoupanos, S., Charalambous, T., Krikidis, I., Petropulu, A., 2021. A survey on reinforcement learning-aided caching in mobile edge networks. arXiv:2105.05564.

Nygren, E., Sitaraman, R.K., Sun, J., 2010. The Akamai network: A platform for high-performance internet applications. SIGOPS Oper. Syst. Rev. 44 (3), 2–19. http://dx.doi.org/10.1145/1842733.1842736.

OpenFog Consortium, 2022a. OpenFog reference architecture for fog computing. https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf. (Accessed 12 July 2022).

Qian, L., Wu, Y., Jiang, F., Yu, N., Lu, W., Lin, B., 2021. NOMA assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial internet of things. IEEE Trans. Ind. Inform. 17 (8), 5688–5698. http://dx.doi.org/10.1109/TII.2020.3001355.

Qian, Y., Wu, J., Wang, R., Zhu, F., Zhang, W., 2019. Survey on reinforcement learning applications in communication networks. J. Commun. Inform. Netw. 4 (2), 30–39. http://dx.doi.org/10.23919/JCIN.2019.8917870.

Qinghua, Z., Ying, C., Jingya, Z., Yong, L., 2020. Computation offloading optimization in edge computing based on deep reinforcement learning. In: 2020 5th International Conference on Mechanical, Control and Computer Engineering. ICMCCE 2020, pp. 1552–1558. http://dx.doi.org/10.1109/ICMCCE51767.2020.00340.

Qiu, X., Zhang, W., Chen, W., Zheng, Z., 2021. Distributed and collective deep reinforcement learning for computation offloading: A practical perspective. IEEE Trans. Parallel Distrib. Syst. 32 (5), 1085–1101. http://dx.doi.org/10.1109/TPDS.2020.3042599.

Qu, C., Calyam, P., Yu, J., Vandanapu, A., Opeoluwa, O., Gao, K., Wang, S., Chastain, R., Palaniappan, K., 2021. DroneCOCoNet: Learning-based edge computation offloading and control networking for drone video analytics. Future Gener. Comput. Syst. 125, 247–262. http://dx.doi.org/10.1016/j.future.2021.06.040.

Ren, T., Niu, J., Dai, B., Liu, X., Hu, Z., Xu, M., Guizani, M., 2021. Enabling efficient scheduling in large-scale UAV-assisted mobile edge computing via hierarchical reinforcement learning. IEEE Internet Things J. http://dx.doi.org/10.1109/JIOT.2021.3071531.

Ren, J., Xu, S., 2021. DDPG based computation offloading and resource allocation for MEC systems with energy harvesting. In: IEEE Vehicular Technology Conference. 2021-April. pp. 1–5. http://dx.doi.org/10.1109/VTC2021-Spring51267.2021.9448922.

Saeik, F., Avgeris, M., Spatharakis, D., Santi, N., Dechouniotis, D., Violos, J., Leivadeas, A., Athanasopoulos, N., Mitton, N., Papavassiliou, S., 2021. Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions. Comput. Netw. 195, 108177. http://dx.doi.org/10.1016/j.comnet.2021.108177.

Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N., 2009. The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. 8 (4), 14–23. http://dx.doi.org/10.1109/MPRV.2009.82.

Seid, A.M., Boateng, G.O., Anokye, S., Kwantwi, T., Sun, G., Liu, G., 2021a. Collaborative computation offloading and resource allocation in multi-UAV-assisted IoT networks: A deep reinforcement learning approach. IEEE Internet Things J. 8 (15), 12203–12218. http://dx.doi.org/10.1109/JIOT.2021.3063188.

Seid, A.M., Boateng, G.O., Mareri, B., Jiang, W., 2021b. Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network. IEEE Trans. Netw. Serv. Manag. http://dx.doi.org/10.1109/TNSM.2021.3096673.

Sha, D., Zhao, R., 2021. DRL-based task offloading and resource allocation in multi-UAV-MEC network with SDN. In: 2021 IEEE/CIC International Conference on Communications in China. ICCC 2021, pp. 595–600. http://dx.doi.org/10.1109/ICCC52777.2021.9580253.

Shahidinejad, A., Farahbakhsh, F., Ghobaei-Arani, M., Malik, M.H., Anwar, T., 2021. Context-aware multi-user offloading in mobile edge computing: a federated learning-based approach. J. Grid Comput. 19 (2), http://dx.doi.org/10.1007/s10723-021-09559-x.

Shakarami, A., Ghobaei-Arani, M., Masdari, M., Hosseinzadeh, M., 2020a. A survey on the computation offloading approaches in mobile edge/cloud computing environment: A stochastic-based perspective. J. Grid Comput. 18, 639–671. http://dx.doi.org/10.1007/s10723-020-09530-2.

Shakarami, A., Ghobaei-Arani, M., Shahidinejad, A., 2020b. A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective. Comput. Netw. 182, 107496. http://dx.doi.org/10.1016/j.comnet.2020.107496.

Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L., 2016. Edge computing: Vision and challenges. IEEE Internet Things J. 3 (5), 637–646. http://dx.doi.org/10.1109/JIOT.2016.2579198.

Shi, S., Wang, M., Gu, S., Zheng, Z., 2021. Energy-efficient UAV-enabled computation offloading for industrial internet of things: a deep reinforcement learning approach. Wirel. Netw. http://dx.doi.org/10.1007/s11276-021-02789-7.

Shi, M., Wang, R., Liu, E., Xu, Z., Wang, L., 2020. Deep reinforcement learning based computation offloading for mobility-aware edge computing. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST. vol. 312 LNICST, pp. 53–65. http://dx.doi.org/10.1007/978-3-030-41114-5_5.

Shu, X., Wu, L., Qin, X., Yang, R., Wu, Y., Wang, D., Liao, B., 2021. Deep reinforcement learning cloud-edge-terminal computation resource allocation mechanism for IoT. Adv. Intell. Syst. Comput. 1274 AISC, 1550–1556. http://dx.doi.org/10.1007/978-981-15-8462-6_177.

Sun, Z., Mo, Y., Yu, C., 2021a. Graph reinforcement learning based task offloading for multi-access edge computing. IEEE Internet Things J. http://dx.doi.org/10.1109/JIOT.2021.3123822.

Sun, Z., Nakhai, M.R., 2020. An online learning algorithm for distributed task offloading in multi-access edge computing. IEEE Trans. Signal Process. 68, 3090–3102. http://dx.doi.org/10.1109/TSP.2020.2991383.

Sun, Z., Zhao, M., Nakhai, M.R., 2021b. Computation offloading in energy harvesting powered MEC network. In: IEEE International Conference on Communications. pp. 1–6. http://dx.doi.org/10.1109/ICC42927.2021.9500984.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction, second ed. The MIT Press.

Tang, M., Wong, V.W., 2020. Deep reinforcement learning for task offloading in mobile edge computing systems. IEEE Trans. Mob. Comput. http://dx.doi.org/10.1109/TMC.2020.3036871.

Tang, D., Zhang, X., Li, M., Tao, X., 2020. Adaptive inference reinforcement learning for task offloading in vehicular edge computing systems. In: 2020 IEEE International Conference on Communications Workshops, ICC Workshops 2020 - Proceedings. pp. 1–6. http://dx.doi.org/10.1109/ICCWorkshops49005.2020.9145133.

Tefera, G., She, K., Chen, M., Ahmed, A., 2020. Congestion-aware adaptive decentralised computation offloading and caching for multiaccess edge computing networks. IET Commun. 14 (19), 3410–3419. http://dx.doi.org/10.1049/iet-com.2020.0630.

Tefera, G., She, K., Shelke, M., Ahmed, A., 2021. Decentralized adaptive resource-aware computation offloading & caching for multi-access edge computing networks. Sustain. Comput. Inform. Syst. 30, http://dx.doi.org/10.1016/j.suscom.2021.100555.

Tong, Z., Deng, X., Ye, F., Basodi, S., Xiao, X., Pan, Y., 2020. Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment. Inform. Sci. 537, 116–131. http://dx.doi.org/10.1016/j.ins.2020.05.057.

Tuong, V.D., Truong, T.P., Nguyena, T.-V., Noh, W., Cho, S., 2021. Partial computation offloading in NOMA-assisted mobile-edge computing systems using deep reinforcement learning. IEEE Internet Things J. 8 (17), 13196–13208. http://dx.doi.org/10.1109/JIOT.2021.3064995.

Tuong, V.D., Truong, T.P., Tran, A.-T., Masood, A., Lakew, D.S., Lee, C., Lee, Y., Cho, S., 2020. Delay-sensitive task offloading for internet of things in nonorthogonal multiple access MEC networks. In: International Conference on ICT Convergence 2020. 2020-October. pp. 597–599. http://dx.doi.org/10.1109/ICTC49870.2020.9289406.

Vahid Dastjerdi, A., Gupta, H., Calheiros, R.N., Ghosh, S.K., Buyya, R., 2016. Fog computing: principles, architectures, and applications. In: Buyya, R., Vahid Dastjerdi, A. (Eds.), Internet of Things. Morgan Kaufmann, pp. 61–75. http://dx.doi.org/10.1016/B978-0-12-805395-9.00004-6.

Vaquero, L.M., Rodero-Merino, L., 2014. Finding your way in the fog: Towards a comprehensive definition of fog computing. SIGCOMM Comput. Commun. Rev. 44 (5), 27–32. http://dx.doi.org/10.1145/2677046.2677052.

Vu, T.T., Huynh, N.V., Hoang, D.T., Nguyen, D.N., Dutkiewicz, E., 2018. Offloading energy efficiency with delay constraint for cooperative mobile edge computing networks. In: 2018 IEEE Global Communications Conference. GLOBECOM, pp. 1–6. http://dx.doi.org/10.1109/GLOCOM.2018.8647856.

Wang, Y., Fang, W., Ding, Y., Xiong, N., 2021. Computation offloading optimization for UAV-assisted mobile edge computing: a deep deterministic policy gradient approach. Wirel. Netw. 27 (4), 2991–3006. http://dx.doi.org/10.1007/s11276-021-02632-z.

Wang, Y., Ge, H., Feng, A., Li, W., Liu, L., Jiang, H., 2020a. Computation offloading strategy based on deep reinforcement learning in cloud-assisted mobile edge computing. In: 2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics. ICCCBDA 2020, pp. 108–113. http://dx.doi.org/10.1109/ICCCBDA49378.2020.9095689.

Wang, X., Guo, H., 2021. Mobility-aware computation offloading for swarm robotics using deep reinforcement learning. In: 2021 IEEE 18th Annual Consumer Communications and Networking Conference. CCNC 2021, pp. 1–4. http://dx.doi.org/10.1109/CCNC49032.2021.9369456.

Wang, H., Ke, H., Liu, G., Sun, W., 2020b. Computation migration and resource allocation in heterogeneous vehicular networks: A deep reinforcement learning approach. IEEE Access 8, 171140–171153. http://dx.doi.org/10.1109/ACCESS.2020.3024683.

Wang, H., Ke, H., Sun, W., 2020c. Unmanned-aerial-vehicle-assisted computation offloading for mobile edge computing based on deep reinforcement learning. IEEE Access 8, 180784–180798. http://dx.doi.org/10.1109/ACCESS.2020.3028553.

Wang, J., Lv, T., Huang, P., Mathiopoulos, P., 2020d. Mobility-aware partial computation offloading in vehicular networks: A deep reinforcement learning based scheme. China Commun. 17 (10), 31–49. http://dx.doi.org/10.23919/JCC.2020.10.003.

Wang, M., Shi, S., Gu, S., Gu, X., Qin, X., 2020e. Q-learning based computation offloading for multi-UAV-enabled cloud-edge computing networks. IET Commun. 14 (15), 2481–2490. http://dx.doi.org/10.1049/iet-com.2019.1184.

Wang, M., Shi, S., Gu, S., Zhang, N., Gu, X., 2020f. Intelligent resource allocation in UAV-enabled mobile edge computing networks. In: 2020 IEEEE 92nd Vehicular Technology Conference. 2020-November. pp. 1–5. http://dx.doi.org/10.1109/VTC2020-Fall49728.2020.9348573.

Wang, D., Tian, X., Cui, H., Liu, Z., 2020g. Reinforcement learning-based joint task offloading and migration schemes optimization in mobility-aware MEC network. China Commun. 17 (8), 31–44. http://dx.doi.org/10.23919/JCC.2020.08.003.

Wang, B., Wang, C., Huang, W., Song, Y., Qin, X., 2020h. A survey and taxonomy on task offloading for edge-cloud computing. IEEE Access 8, 186080–186101. http://dx.doi.org/10.1109/ACCESS.2020.3029649.

Wang, Z., Zhu, Q., 2020. Partial task offloading strategy based on deep reinforcement learning. In: 2020 IEEE 6th International Conference on Computer and Communications. ICCC 2020, pp. 1516–1521. http://dx.doi.org/10.1109/ICCC51575.2020.9345003.

Wei, D., Ma, J., Luo, L., Wang, Y., He, L., Li, X., 2021. Computation offloading over multi-UAV MEC network: A distributed deep reinforcement learning approach. Comput. Netw. 199, http://dx.doi.org/10.1016/j.comnet.2021.108439.

Wu, S., Sun, Q., Zhou, A., Wang, S., Lei, T., 2020. Adaptive edge resource allocation for maximizing the number of tasks completed on time: A deep Q-learning approach. Commun. Comput. Inf. Sci. 1267, 355–367. http://dx.doi.org/10.1007/978-981-15-9213-3_28.

Wu, Z., Yan, D., 2021. Deep reinforcement learning-based computation offloading for 5G vehicle-aware multi-access edge computing network. China Commun. 18 (11), 26–41. http://dx.doi.org/10.23919/JCC.2021.11.003.

Xiao, A., Chen, H., Wu, S., Ma, L., Zhou, F., Ma, D., 2021. Dynamic priority-based computation offloading for integrated maritime-satellite mobile networks. Commun. Comput. Inf. Sci. 1353 CCIS, 70–83. http://dx.doi.org/10.1007/978-981-16-1967-0_5.

Xu, S., Guo, C., Hu, R.Q., Qian, Y., 2021. BlockChain inspired secure computation offloading in a vehicular cloud network. IEEE Internet Things J. http://dx.doi.org/10.1109/JIOT.2021.3054866.

Xu, D., Li, Y., Chen, X., Li, J., Hui, P., Chen, S., Crowcroft, J., 2018. A survey of opportunistic offloading. IEEE Commun. Surv. Tutor. 20 (3), 2198–2236. http://dx.doi.org/10.1109/COMST.2018.2808242.

Xu, S., Liu, Q., Gong, B., Qi, F., Guo, S., Qiu, X., Yang, C., 2020. RJCC: Reinforcement-learning-based joint communicational-and-computational resource allocation mechanism for smart city IoT. IEEE Internet Things J. 7 (9), 8059–8076. http://dx.doi.org/10.1109/JIOT.2020.3002427.

Yan, J., Bi, S., Zhang, Y.J.A., 2020. Offloading and resource allocation with general task graph in mobile edge computing: A deep reinforcement learning approach. IEEE Trans. Wireless Commun. 19 (8), 5404–5419. http://dx.doi.org/10.1109/TWC.2020.2993071.

Yang, G., Hou, L., Cheng, H., He, X., He, D., Chan, S., 2020a. Computation offloading time optimisation via Q-learning in opportunistic edge computing. IET Commun. 14 (21), 3898–3906. http://dx.doi.org/10.1049/iet-com.2020.0765.

Yang, L., Zhang, H., Li, M., Guo, J., Ji, H., 2018. Mobile edge computing empowered energy efficient task offloading in 5G. IEEE Trans. Veh. Technol. 67 (7), 6398–6409. http://dx.doi.org/10.1109/TVT.2018.2799620.

Yang, M., Zhu, H., Wang, H., Koucheryavy, Y., Samouylov, K., Qian, H., 2020b. Peer to peer offloading with delayed feedback: An adversary bandit approach. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. 2020-May. pp. 5035–5039. http://dx.doi.org/10.1109/ICASSP40776.2020.9053680.

Yang, M., Zhu, H., Wang, H., Koucheryavy, Y., Samouylov, K., Qian, H., 2021. An online learning approach to computation offloading in dynamic fog networks. IEEE Internet Things J. 8 (3), 1572–1584. http://dx.doi.org/10.1109/JIOT.2020.3015522.

Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., Jue, J.P., 2019. All one needs to know about fog computing and related edge computing paradigms: A complete survey. J. Syst. Archit. 98, 289–330. http://dx.doi.org/10.1016/j.sysarc.2019.02.009.

Yu, S., Chen, X., Zhou, Z., Gong, X., Wu, D., 2021. When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network. IEEE Internet Things J. 8 (4), 2238–2251. http://dx.doi.org/10.1109/JIOT.2020.3026589.

Yuan, X., Zhu, Y., Zhao, Z., Zheng, Y., Pan, J., Liu, D., 2020. An A3C-based joint optimization offloading and migration algorithm for SD-WBANs. In: 2020 IEEE Globecom Workshops, GC Wkshps 2020 - Proceedings. pp. 1–6. http://dx.doi.org/10.1109/GCWkshps50303.2020.9367507.

Zamzam, M., Elshabrawy, T., Ashour, M., 2019. Resource management using machine learning in mobile edge computing: A survey. In: 2019 Ninth International Conference on Intelligent Computing and Information Systems. ICICIS, pp. 112–117. http://dx.doi.org/10.1109/ICICIS46948.2019.9014733.

Zhan, Y., Guo, S., Li, P., Zhang, J., 2020a. A deep reinforcement learning based offloading game in edge computing. IEEE Trans. Comput. 69 (6), 883–893. http://dx.doi.org/10.1109/TC.2020.2969148.

Zhan, W., Luo, C., Wang, J., Wang, C., Min, G., Duan, H., Zhu, Q., 2020b. Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing. IEEE Internet Things J. 7 (6), 5449–5465. http://dx.doi.org/10.1109/JIOT.2020.2978830.

Zhang, K., Cao, J., Liu, H., Maharjan, S., Zhang, Y., 2020a. Deep reinforcement learning for social-aware edge computing and caching in urban informatics. IEEE Trans. Ind. Inform. 16 (8), 5467–5477. http://dx.doi.org/10.1109/TII.2019.2953189.

Zhang, J., Du, J., Jiang, C., Shen, Y., Wang, J., 2020b. Computation offloading in energy harvesting systems via continuous deep reinforcement learning. In: IEEE International Conference on Communications. 2020-June. pp. 1–6. http://dx.doi.org/10.1109/ICC40277.2020.9148938.

Zhang, J., Du, J., Shen, Y., Wang, J., 2020c. Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach. IEEE Internet Things J. 7 (10), 9303–9317. http://dx.doi.org/10.1109/JIOT.2020.3000527.

Zhang, J., Du, J., Wang, J., Shen, Y., 2020d. Hybrid decision based deep reinforcement learning for energy harvesting enabled mobile edge computing. In: 2020 International Wireless Communications and Mobile Computing. IWCMC 2020, pp. 2100–2105. http://dx.doi.org/10.1109/IWCMC48107.2020.9148398.

Zhang, X., Lin, W., Li, Y., Cui, Q., Tao, X., Huang, X., Ren, P., 2020e. Moving server: Follow-up computation offloading paradigm for vehicular users. In: 2020 IEEE/CIC International Conference on Communications in China. ICCC 2020, pp. 226–231. http://dx.doi.org/10.1109/ICCC49849.2020.9238895.

Zhang, Y., Liu, T., Zhu, Y., Yang, Y., 2020f. A deep reinforcement learning approach for online computation offloading in mobile edge computing. In: 2020 IEEE/ACM 28th International Symposium on Quality of Service. IWQoS 2020, pp. 1–10. http://dx.doi.org/10.1109/IWQoS49365.2020.9212868.

Zhang, L., Luo, J., Gao, L., Zheng, F.-C., 2020g. Learning-based computation offloading for edge networks with heterogeneous resources. In: IEEE International Conference on Communications. 2020-June. pp. 1–6. http://dx.doi.org/10.1109/ICC40277.2020.9149171.

Zhang, X., Shen, Y., Yang, B., Zang, W., Wang, S., 2021a. DRL based data offloading for intelligent reflecting surface aided mobile edge computing. In: IEEE Wireless Communications and Networking Conference. 2021-March. WCNC, pp. 1–7. http://dx.doi.org/10.1109/WCNC49053.2021.9417469.

Zhang, J., Shi, W., Zhang, R., Liu, W., 2021b. Computation offloading and shunting scheme in wireless wireline internetwork. IEEE Trans. Commun. http://dx.doi.org/10.1109/TCOMM.2021.3092414.

Zhang, J., Shi, W., Zhang, R., Liu, S., 2021c. Deep reinforcement learning for offloading and shunting in hybrid edge computing network. In: 2021 IEEE International Conference on Communications Workshops, ICC Workshops 2021 - Proceedings. pp. 1–6. http://dx.doi.org/10.1109/ICCWorkshops50388.2021.9473628.

Zhang, L., Xu, J., 2020. Fooling edge computation offloading via stealthy interference attack. In: Proceedings - 2020 IEEE/ACM Symposium on Edge Computing, SEC 2020. pp. 415–419. http://dx.doi.org/10.1109/SEC50012.2020.00062.

Zhang, H., Yang, Y., Huang, X., Fang, C., Zhang, P., 2021d. Ultra-low latency multi-task offloading in mobile edge computing. IEEE Access 9, 32569–32581. http://dx.doi.org/10.1109/ACCESS.2021.3061105.

Zhang, H., Yu, T., 2020. Taxonomy of reinforcement learning algorithms. In: Dong, H., Ding, Z., Zhang, S. (Eds.), Deep Reinforcement Learning: Fundamentals, Research and Applications. Springer Singapore, pp. 125–133. http://dx.doi.org/10.1007/978-981-15-4095-0_3.

Zhang, L., Zhang, Z.-Y., Min, L., Tang, C., Zhang, H.-Y., Wang, Y.-H., Cai, P., 2021e. Task offloading and trajectory control for UAV-assisted mobile edge computing using deep reinforcement learning. IEEE Access 9, 53708–53719. http://dx.doi.org/10.1109/ACCESS.2021.3070908.

Zheng, X., Li, M., Chen, Y., Guo, J., Alam, M., Hu, W., 2021. Blockchain-based secure computation offloading in vehicular networks. IEEE Trans. Intell. Transp. Syst. 22 (7), 4073–4087. http://dx.doi.org/10.1109/TITS.2020.3014229.

Zhou, H., Jiang, K., Liu, X., Li, X., Leung, V.C., 2021. Deep reinforcement learning for energy-efficient computation offloading in mobile edge computing. IEEE Internet Things J. http://dx.doi.org/10.1109/JIOT.2021.3091142.

Zhu, S., Gui, L., Cheng, N., Zhang, Q., Sun, F., Lang, X., 2020. UAV-enabled computation migration for complex missions: A reinforcement learning approach. IET Commun. 14 (15), 2472–2480. http://dx.doi.org/10.1049/iet-com.2019.1188.

Zhu, X., Luo, Y., Liu, A., Bhuiyan, M.Z.A., Zhang, S., 2021. Multiagent deep reinforcement learning for vehicular computation offloading in IoT. IEEE Internet Things J. 8 (12), 9763–9773. http://dx.doi.org/10.1109/JIOT.2020.3040768.

Diego Hortelano received his Ph.D. in low-power wireless communication networks from the University of Castilla-La Mancha, Spain, in 2021. In the same year he joined the Optical Communications Group, Universidad de Valladolid, as a Postdoctoral Researcher. He is currently an assistant professor in the Department of Computer Science and Statistics, Universidad Rey Juan Carlos. His main research fields are Internet of Things, wireless communication protocols, edge computing and the application of artificial intelligence to these areas.

Ignacio de Miguel received the degree in telecommunication engineering and the Ph.D. degree from the Universidad de Valladolid (UVa), Spain, in 1997 and 2002, respectively. Since 1997, he has worked as a Lecturer at UVa, and is currently an Associate Professor. He is also the Coordinator of the master's degree in telecommunication engineering and the master's degree in big data science at UVa. He has also been a Visiting Research Fellow at University College London, U.K. His main research interests include the design, control and performance evaluation of optical networks, edge computing, and the application of artificial intelligence techniques in those environments. He has been a member of the TPC of several international conferences, besides being the Chair of the TPC and the Local Organising Committee of NOC 2009. He was a recipient of the Nortel Networks Prize to the Best Ph.D. Thesis on Optical Internet in 2002, awarded by the Spanish Institute and the Association of Telecommunication Engineers (COIT/AEIT).

Ramón J. Durán received the degree in telecommunication engineering and the Ph.D. degree from the University of Valladolid, Spain, in 2002 and 2008, respectively. He currently works as an Associate Professor with the Universidad de Valladolid. He is also the Coordinator of the Spanish Research Thematic Network "Go2Edge: Engineering Future Secure Edge Computing Networks, Systems and Services" composed of 15 entities and the H2020 IoTalentum Project. He has authored more than 100 papers in international journals and conferences. His current research interests include the use of artificial intelligence techniques for the design, optimisation, and operation of future heterogeneous networks, multi-access edge computing, and network function virtualization.

Juan Carlos Aguado received the degree in telecommunication engineering and the Ph.D. degree from Universidad de Valladolid, Spain, in 1997 and 2005, respectively. Since 1998, he has been working as a Junior Lecturer with the Universidad de Valladolid, where he is currently an Associate Professor. He has also been a Postdoctoral Researcher with the Group of Transmisiones Ópticas de Banda Ancha (TOyBA), University of Zaragoza. His current research interests include vehicular communications and networking, and the application of edge computing and artificial intelligence techniques to these environments, as well as the design and performance evaluation of optical networks.

Noemí Merayo received the degree in telecommunication engineering from Universidad de Valladolid, Spain, in February 2004, and the Ph.D. degree from the Optical Communication Group, Universidad de Valladolid, in July 2009. Since 2005, she has been working as a Lecturer with the Universidad de Valladolid. She has also been a Visiting Research Fellow with the Optical Networks Group, Science and Technology Research Institute (STRI), University of Hertfordshire, another at the TOyBA Research Group, University of Zaragoza, and more recently at the Technical University of Munich (TUM). She is currently coordinating the master's degree in physics and technology of lasers at the University of Valladolid and the University of Salamanca. Her research interests include the design and performance evaluation of passive optical networks, and the application of artificial intelligence techniques to communication networks and edge computing systems.

Lidia Ruiz received the degree in telecommunication engineering, the M.Res. degree in information and telecommunication technologies, and the Ph.D. degree from the Universidad de Valladolid, Spain, in 2013, 2015, and 2020, respectively. She has been a Visiting Researcher with the Politecnico di Milano, Italy, and has worked as an IT Consultant. She is currently working as a Postdoctoral Researcher. Her research interests include 5G, network function virtualization, edge computing and optical networking.

Adrian Asensio received a M.Sc. in Telecommunications Engineering and a Ph.D. degree in Computer Science from the Universitat Politècnica de Catalunya (UPC). His publications include book chapters, papers in relevant international journals and papers in international refereed conferences. He is with the CRAAX lab since 2017. His current research interests include mathematical models and algorithms design and their applications in network management, and in cloud and edge computing.

Xavi Masip-Bruin received the M.Sc. and Ph.D. degrees in Telecommunications Engineering from the Universitat Politècnica de Catalunya (UPC). He is currently a Full Professor with the Computer Science Department, UPC, where he is also serving as the Director of the CRAAX Laboratory. In 2011 and 2012, he was a Visitor Professor with UPEC, Paris. His current research interests include cloud and fog computing, network management, cybersecurity, the IoT, and particularly on analysing the benefits brought by combining fog and cloud paradigms. He has been involved in many different research initiatives at national and international level as well as in many contracts with the industry. He has been recognised with the 2016 IBM Faculty Award.

Patricia Fernández received her Telecommunication Engineer Degree from the Universitat Politècnica de Catalunya, BarcelonaTech (UPC), Spain, in 1997. After that, she joined the Optical Communications Group of the Universidad de Valladolid where she obtained her PhD Degree in 2004. She is author of more than 100 papers in international journals and conferences. Currently, she is a Professor and she was the Head of the Technical School of Telecommunications Engineering at Universidad de Valladolid.

Rubén M. Lorenzo received his Telecommunication Engineer and PhD degrees from the Universidad de Valladolid, Spain, in 1996 and 1999, respectively. From 1996 to 2000, he was a Junior Lecturer at the Universidad de Valladolid and joined the Optical Communications Group. Since 2000, he has been a Lecturer. His research interests include optical networks and multi-access edge computing. He was the Head of the Technical School of Telecommunications Engineering at Universidad de Valladolid and Research Director of CEDETEL (Center for the Development of Telecommunications in Castilla y León).

Evaristo J. Abril received the degree in telecommunication engineering and the Ph.D. degree from the Universidad Politècnica de Madrid, Spain, in 1985 and 1987, respectively. From 1984 to 1986, he was a Research Assistant with the Universidad Politècnica de Madrid and became a Lecturer in 1987. Since 1995, he has been a Full Professor with the Universidad de Valladolid, Spain, where he founded the Optical Communications Group. His research interests include integrated optics, optical communication systems and networks, vehicular communications and edge computing. He has authored more than 100 papers in international journals and conferences.