Engineering Applications of Artificial Intelligence 95 (2020) 103840

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Computation offloading in Edge Computing environments using Artificial Intelligence techniques☆

Gonçalo Carvalho [a,*], Bruno Cabral [a], Vasco Pereira [a], Jorge Bernardino [b,a]

[a] Univ Coimbra, Centre for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering, Portugal
[b] Polytechnic of Coimbra - ISEC, Portugal

ARTICLE INFO

Keywords:
Artificial Intelligence
Computation offloading
Edge Computing
Machine Learning

ABSTRACT

Edge Computing (EC) is a recent architectural paradigm that brings computation close to end-users with the aim of reducing latency and bandwidth bottlenecks, which 5G technologies are committed to further reduce, while also achieving higher reliability. EC enables computation offloading from end devices to edge nodes. Deciding whether a task should be offloaded, or not, is not trivial. Moreover, deciding when and where to offload a task makes things even harder and making inadequate or off-time decisions can undermine the EC approach. Recently, Artificial Intelligence (AI) techniques, such as Machine Learning (ML), have been used to help EC systems cope with this problem. AI promises accurate decisions, higher adaptability and portability, thus diminishing the cost of decision-making and the probability of error. In this work, we perform a literature review on computation offloading in EC systems with and without AI techniques. We analyze several AI techniques, especially ML-based, that display promising results, overcoming the shortcomings of current approaches for computing offloading coordination We sorted the ML algorithms into classes for better analysis and provide an in-depth analysis on the use of AI for offloading, in particular, in the use case of offloading in Vehicular Edge Computing Networks, actually one technology that gained more relevance in the last years, enabling a vast amount of solutions for computation and data offloading. We also discuss the main advantages and limitations of offloading, with and without the use of AI techniques.

1. Introduction

The main quality of end devices, such as smartphones, tablets, and notebooks, is their mobility. These are increasingly omnipresent in our daily lives as convenient tools for communication, entertainment, business, social networking, among others. Furthermore, emerging mobile applications typically require intensive computation and high energy consumption, which end devices are not able to cope with (Cao and Cai, 2018). However, Cloud systems mitigate this problem at the price of higher latency and increased data communication.

To overcome this problem, Edge Computing (EC) presents itself as a computing paradigm that brings Cloud resources closer to end-users, to the edge of the network, in order to minimize some of the Cloud Computing problems. EC was a term first coined by Akamai, in the late 1990s, describing the architecture of content delivery networks (Li et al., 2018a). Additionally, EC enables local data processing by taking advantage of nearby devices.

There are four main architectures regarding the EC paradigm. A brief definition of each will be provided next.

Multi-access Edge Computing (MEC), according to the European Telecommunication Standards Institute (ETSI) (Patel et al., 2014), started to be named: "*Mobile Edge Computing, and was defined as an Information Technology service environment and Cloud Computing capabilities at the edge of the mobile network, within the Radio Access Network (RAN) and in proximity to mobile subscribers*". However, since September 2016, the same Institute encompassed the non-mobile features of the architecture, including Wi-Fi and fixed access technologies, Virtual Network Functions and Software Defined Networking, as well as other virtualization technologies apart from virtual machines, resulting in the modification of name from "Mobile" to "Multi-access" (Mouradian et al., 2018; Roman et al., 2018).

Fog Computing (FC) is defined by Cisco (Bonomi et al., 2012) as "*a highly virtualized platform that provides computation, storage, and networking services between end devices and traditional Cloud Computing data centers, typically, but not exclusively located at the edge of network*". Fog Computing arose in 2010 to cope with the vast amount of Internet of Things devices and Big Data, as an extension of Cloud Computing and designed for real-time low-latency applications.

The **Cloudlet** concept and architecture were proposed by Satyanarayanan et al. (2009), and defined as "*a trusted, resource-rich computer or cluster of computers that are well-connected to the Internet and available*

☆ This document is the results of the research project funded by InfraCrit/Mobiwise/Mobilizador 5G.
* Corresponding author.
E-mail addresses: gcarvalho@dei.uc.pt (G. Carvalho), bcabral@dei.uc.pt (B. Cabral), vasco@dei.uc.pt (V. Pereira), jorge@isec.pt (J. Bernardino).

https://doi.org/10.1016/j.engappai.2020.103840
Received 23 December 2019; Received in revised form 9 April 2020; Accepted 20 July 2020
Available online xxxx
0952-1976/

for use by nearby end devices, offering computing and storage resources". Cloudlets aim to endorse flexibility, mobility, scalability, and elasticity. Thus, they are discoverable, generic, stateless servers located in single-hop proximity of end devices, benefiting from virtual machines able to operate disconnected (Dastjerdi et al., 2016). Acting as a data center in a box and through a high-bandwidth network, Cloudlets resemble a set of virtualized high-performance computers (Li et al., 2018a).

The term **Mobile Cloud Computing (MCC)** according to the Mobile Cloud Computing Forum, referenced by Yi et al. (2015) is defined as, "*an infrastructure where both the data storage and data processing happen outside of the mobile device. Mobile Cloud applications move the computing power and data storage away from mobile phones and into the Cloud, bringing applications and Mobile Computing to not just smartphone users but a much broader range of mobile subscribers*".

One way to explore EC systems is for computation offloading into the edge devices, which has already been successfully implemented as an enabler of resource-intensive applications on end devices (e.g. MAUI Cuervo et al., 2010 and Cuckoo Kemp et al., 2012). Offloading is a solution to increase mobile systems' capabilities by migrating computation to more resourceful devices, located nearby, such as edge nodes, fog nodes, Cloudlets, base stations or access points. In addition, by remotely running applications, or parts of it, the end devices' battery lifetime will be longer (Huang et al., 2012). It is also commonly used by single users to enhance the insufficient computation resources of their processing device (Enzai and Tang, 2014). Khan et al. (2014) defined computation offloading as the technique where resource-intensive computations are migrated from a end device to the resource-rich infrastructures, such as Cloud or nearby edge devices. According to Kumar et al. (2013), "cyber foraging" and "surrogate computing" are other terms by which computation offloading can also be addressed.

To successfully achieve the benefits of processing power and energy-saving which offloading aims, the following features must be available: heavy computation resources, fast server, small data exchange, and high bandwidth (Kumar et al., 2013). Fig. 1 depicts the computation offloading process. The workflow begins with the execution of an application: if the application supports offloading, its use is beneficial and the resources needed are available at that moment, then offloading is executed. Otherwise, the application is processed locally in the device. If the application is able to find resources to perform offloading tasks in EC systems, then it will continue to run on remote resources, instead of on-device processing, thus achieving a successful offloading process.

The decision for offloading a task is not trivial, and the questions "why offload?", "what to offload?", "when should the task be offloaded?", "how to manage the user mobility?", and "where to offload the task?", makes the offloading decision even harder. Khan et al. (2014) added to this complexity the different elements that are involved in the computation offloading process, such as user, connectivity, smartphone, application model, application, and the resource-rich infrastructure. La and Kim (2014) introduced five criteria for defining the offloading schemes, however, Enzai and Tang (2014) discussed six. The following criteria are a combination of these two offloading proposals:

- *Objectives*: minimizing execution time and energy consumption;
- *The granularity of components to offload*: full or partial offloading of the task, thread, application or program;
- *Scheme*: static or dynamic;
- *Timing of transmitting components*: pre-deployment, on-demand transmission;
- *Route of transmitting components*: direct or indirect;
- *Communication*: client to server or server to server;
- *Determination of offloading server*: pre-determined or dynamically discovered;
- *Adaptation*: contemplating the different contexts or instances of task execution.

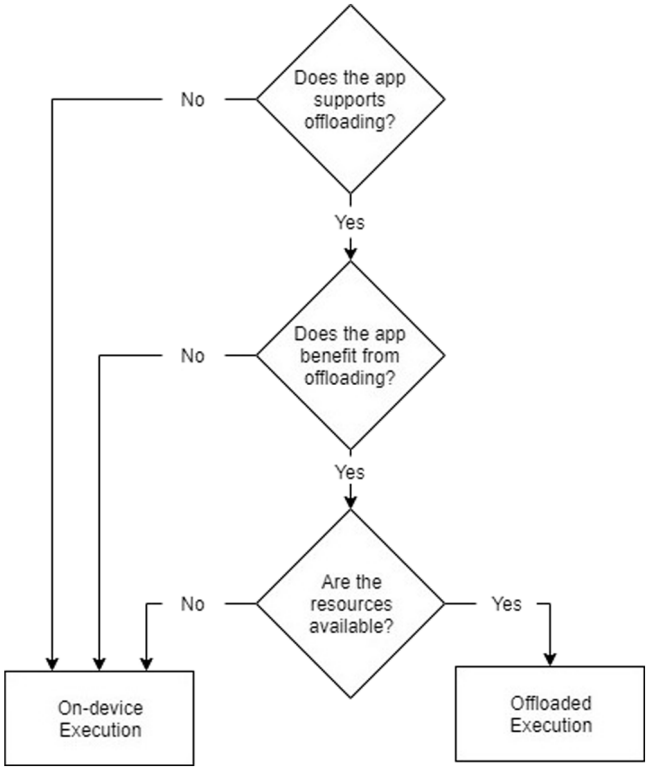

Fig. 1. Computation offloading process.

Offloading in EC environments is challenging. First, offloading may not always achieve the lowest cost due to possible high communication and remote execution costs that may also change with time or available connections. Therefore, the software developer must always evaluate costs when deciding whether to execute a job locally, on the end device, or offloading to nearby cloudlets or other edge nodes. Thus, a dynamic decision-making algorithm is necessary. Second, the user's mobility is an issue because their end devices can become disconnected from the edge node to which it is offloading, preventing the user from receiving the result, causing a failure. Moreover, there is a need to develop admission policies for edge nodes so end-users can access these devices and proceed with the offloading requests. Furthermore, the fluctuations in Internet connection quality need to be accounted for (Zhang et al., 2015). Liu et al. (2019) described yet another challenge, the energy consumption spent in the offloading decision. Therefore, the offloading process, due to its multiplicity of variables, is ideal for the implementation of Artificial Intelligence (AI) techniques.

AI techniques are becoming one of the main trends in computation offloading coordination, essentially by the rising number of research papers published in this area. These techniques will enable efficient access to massive amounts of distributed data over edge devices. EC devices are evolving in what concerns processing power, energy consumption, data storage, and memory capacity, creating space for on-device Machine Learning (ML) (Yazici, 2018). Because network environments are unstable and its parameters or requirements may suddenly change, AI techniques can provide for intelligent on-time reconfiguration. 5G infrastructures will allow EC systems to better manage computation offloading by diminishing link costs, guaranteeing higher bandwidths, and enabling massive scale connections, enforcing decentralization, and trimming communication with Cloud data centers (Kiss et al., 2018). Also, achieving an efficient decision for the offloading process in future 5G environments, will improve scenarios such as smart cities, smart transportation, augmented reality, edge analytics, and image/video processing. Moreover, every prospective

computation offloading application will benefit or necessarily have to consider 5G features.

In this work, we aim to provide an exhaustive literature review on the application of AI methods in computation offloading. The main contributions of this paper are the following:

- We provide a literature review on computation offloading in EC systems without AI and the enhancements that could be achieved by applying AI techniques.
- We provide a comparative analysis of AI techniques used in computation offloading coordination in EC technologies.
- We provide an in-depth analysis of the use of AI for offloading, in particular, the use case of Vehicular Edge Computing Network.
- We provide a discussion on offloading to edge/Cloud, using or not the AI approaches, and on the main advantages and limitations of the ML algorithms.

The rest of this paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we offer background knowledge of EC and offloading methods, identifying some limitations. In Section 4, we describe current AI approaches to EC computing offloading, and analyze in detail the use case of offloading in Vehicular Edge Computing Networks. In Section 5, we provide a discussion on the main advantages and limitations of offloading, with and without the use of AI. Finally, Section 6 concludes the paper and presents future work.

2. Related work

Concerning related work, despite the rise in the number of published works on computation or data offloading, the amount of works surveying this topic is still limited. Furthermore, addressing the use of AI in EC environments is a quite recent approach.

In **2013**, Kumar et al. (2013) provided a survey as an overview of the background, motivations, techniques, systems, technological enablers, architectures and research areas for offloading computation, aiming to allow readers to get acquainted with computation offloading for mobile systems. They also provide some insights on the offloading future which will rely on sensor deployment, that produce huge amounts of data but have limited computing resources, and the evolution of smartphones as the main computation device.

Dinh et al. (2013) addressed the issues in Mobile Cloud Computing and specifically computation offloading in terms of static and dynamic environments.

Fernando et al. (2013) discussed the major approaches in computation offloading methods, such as client–server communication, virtualization, and mobile agents, and discuss the advantages and weaknesses of each solution. The authors mention the importance to find an answer to how to perform offloading efficiently, and what incentives can be provided for devices that share their resources. Some other challenges are provided, such as virtualization techniques, which also gain from the standardization of mobile systems, which are still highly heterogeneous, and the difficulty to compare studies on energy saving offloading techniques since it also depends on the application itself.

In **2014**, Rahimi et al. (2014) surveyed computation offloading as the method to alleviate resource limitations in Mobile Cloud Computing and also provide open research issues, such as joint optimization efforts amongst devices, pricing an offloading service is still not defined, and studies of effectiveness of client/server or virtualization-based architectures to run different mobile applications.

In **2015**, Wang et al. (2015) discussed the challenges and future direction of computation offloading by presenting the existing solution in terms of static partitioning, dynamic profiling, and the offloading decision. The authors highlight the importance of application partitioning and proceed to mention the necessity for automatic code/computation offloading since it is still human-based.

In **2016**, Ahmed and Ahmed (2016) surveyed Mobile Edge Computing and discuss several frameworks, algorithms or scenarios where computation offloading is used. They state offloading should be performed in edge devices, keeping low latency, when compared to Cloud offloading methods, and reliability, even when a node leaves the network, is a feature that must be integrated on the offloading process.

Pang et al. (2016) developed their work regarding Cloudlets and the computation and data offloading, where the first was analyzed regarding the objective, underlying technology, decision and partition granularity, comparing the approach where a Cloudlet node is included to other offloading frameworks without this middle layer node. The authors also introduce two interesting research directions, such as creating a cluster of Cloudlets to aid every node with the produced data, restraining the need for data to travel to the core of the network, and the use of Cloudlets to perform social media offloading, keeping content and services in the edge, diminish security and privacy risks.

In **2017**, Mach and Becvar (2017) described Multi-access Edge Computing benefits in computation offloading and survey the research in three main extents, the decision on offloading, the computing resource allocation, and mobility management. This detailed survey introduces open issues on the distribution and management of resources, the difficulties around the offloading decision, allocation of computing resources, mobility management, since mobile offloading may occur while the user is moving, and traffic management considering also conventional data with offloaded data.

Wang et al. (2017) added a literature review on the research efforts on computation offloading regarding the following five scenarios: single-user systems, multi-user systems, offloading to Multi-access Edge Computing servers, offloading to other devices and mobility awareness. Nonetheless, the authors state that cooperation between the edge and the core is crucial to improve overall system capacity, and the integration of computing, storage and communication will allow optimal performance, but it is still an open issue.

In **2018**, Aazam et al. (2018) presented offloading in Fog Computing environments regarding eight criteria used in the offloading process: excessive computation or resource constraint; latency; load balancing; permanent or long-term storage; data management and organization; privacy and security; accessibility; and affordability, feasibility, and maintenance. Then proceed to introduce the enabling technologies, such as wireless technology, smart, intelligent, and autonomous applications, virtualization and containment, and parallelism. The scenarios which offloading is fundamental, and future research challenges in offloading, such as fault-tolerance and reconfiguration of the offloading system, incentives, monitoring and scalability amongst others already mentioned in previous works.

In **2019**, Fei et al. (2019) explored how ML methods should be deployed and integrated into Cloud and Fog architectures for better fulfillment of mission-critical and time-critical requirements arising from cyber–physical systems, but lack to explore computation offloading as an enhancement of their system.

Cao et al. (2019) addressed intelligent offloading in Mobile Edge Computing (Multi-access Edge Computing). The authors detailed some applications, such as video acceleration, augmented reality, connected vehicles, and stream analysis. These applications would benefit from using ML-based approaches, which the authors started to divide into two categories by learning style: Reinforcement Learning and Supervised with Unsupervised Learning. However the remaining two categories, Deep Learning and Deep Reinforcement Learning are types of algorithms that are contained within the three main learning styles, thus their division is not appropriate. Nevertheless, the authors approach different algorithms for each learning style, provide the best application for each and survey works using each algorithm, such as Markov Decision Process and Q-Learning (Reinforcement Learning), Support Vector Machine and Support Vector Regression (Supervised), K-means (Unsupervised Learning), Multi-layer Neural Networks (Deep Learning), and Deep Q-Learning (Deep Reinforcement Learning). Moreover, some

advantages and problems using AI in EC environments were discussed, highlighting the comparison between scenarios with and without ML. Finally, they presented an intelligent offloading framework, which observes-orients-decides-acts with an evaluation system for continuous improvement of the decision process loop.

Zhou et al. (2019) surveyed Edge Intelligence, focusing on Deep Neural Networks. The authors provide a brief introduction to AI, although mixing Deep Neural Networks and Deep Reinforcement Learning models. The authors add four benefits between EC and AI, such as edge data needs AI to achieve its full potential, AI can prosper with EC data and application scenarios, the wide use of AI requires EC as a fundamental infrastructure, and EC can gain popularity with AI applications. The authors also presented a six-level division for Edge Intelligence, depending on the amount and path length of data offloading. From the closest to the Cloud – Cloud–Edge Coinference and Cloud Training – to the closest to the edge — All On-Device. This path leads to minor transmission latency of data offloading and bandwidth costs, and increased data privacy. Concerning training architectures, the authors, consider three possibilities, centralized (just Cloud), decentralized (on edge devices), and hybrid (Cloud–edge devices). For the Deep Neural Networks model inference, the authors considered four architectures: edge-based, device-based, edge-device, and edge-Cloud. Regarding offloading, the authors consider this technique to partition the model in order to reduce the weight of Edge Intelligence applications execution on end devices. From the several future research directions mentioned, offloading integrates the analysis on efficient service discovery and resource management. Despite being a comprehensive survey on just one group of AI techniques Deep Learning, offloading is also just superficially mentioned.

Deng et al. (2019) also surveyed Edge Intelligence as an integration between EC and AI. Furthermore, the authors categorize their approach in AI for Edge and AI on Edge, considering that both should enhance Quality of Experience for the end user. The first focus on granting better solutions to optimizations problems. The authors presented several research areas which where divided into Topology, Content, and Service (Computation Offloading is placed within the Service area since AI delivers effective tools for solving complex learning, planning, and decision-making problems). The latter, researches the implementation of AI models processes on the edge, regarding the following research areas: Model Adaptation, Framework Design, and Processor Acceleration. In addition, the authors mention challenges for each branch of Edge Intelligence: model establishment, algorithm deployment and balance between optimality and efficiency are connected to AI for Edge. Data availability, model selection and coordination mechanism are associated challenges for AI on Edge.

Chen and Ran (2019) presented a review at the intersection of Deep Learning and EC, focused on the applications, training methods and challenges. The authors state that applying Deep Learning at the edge of the network will tackle the latency, scalability and privacy challenges associated with centralized approaches. The authors surveyed methods for fast inference, which were divided into three architectures: on-device computation, edge server computation, and computing across edge devices. Offloading was addressed in the last two architectures. In edge server computation the authors found existing works on offloading data. In computing across edge devices the analysis is performed under four scenarios: binary offloading (offloading the entire Deep Neural Network or not), partial offloading (only part of the computations are offloaded), hierarchical offloading (combining edge devices, edge servers, and Cloud in the offloading process), and distributed computing (the Deep Neural Network is simply distributed and not offloaded). In these scenarios, the decision aims to reach the best trade-off between energy consumption, accuracy, latency and input size for the models. The authors also reviewed a recent approach on Deep Learning at the edge which consists on training the models on edge devices. Regarding the challenges of offloading Deep Neural Networks, the authors mention the continuous evolution of the models, which may be unfitted to be partial offloaded, increasing the difficulty of the offloading decision process.

From our bibliographic research, prior to 2019, no studies addressed the use of AI algorithms for computation offloading. However, as a result of the improvements of edge devices computation capabilities, deploying AI, and more especially ML algorithms became feasible. Thus, it excludes the need to regularly use Cloud services to benefit from the ML models. Furthermore, the offloading process, including the decision, achieves high value with this recent approach in these stochastic environments.

3. Fundamentals of computation offloading in edge computing

Distributed Computing paradigms have been continuously evolving. Recently, Cloud Computing introduced new features such as on-demand services and applications, cost-effective, scalable, and efficient data storage and management (Hall and Miller, 2018), fault-tolerance, simple management, fair pricing policies, among others, resulting in worldwide availability (Mouradian et al., 2018). However, the technological progress and the quick rising of the amount of data generated, processed and stored, brought new problems to Cloud Computing. For example, undesirable latency in accessing Cloud servers, lack of mobility support, the ineffectiveness of context awareness, overhead in data transmission (Li et al., 2018a), difficulties to assure privacy, removal of systems control from the users to the Cloud (Garcia Lopez et al., 2015), limitations on location and connectivity between Cloud and end devices (Mouradian et al., 2018).

To tackle these challenges, a new computing paradigm was created, known as EC, not to replace Cloud Computing, but as its complement. Moreover, because EC processes data locally, benefiting from local devices, it surpasses mere data migration or computation offloading from the Cloud, by encompassing latency-sensitive, geo-distributed and mobile applications, as well as distributed control systems (Li et al., 2018a). Furthermore, EC systems aim to drive computation proficiency to the edge of the network, lower the restrictions of a centralized architecture, such as implementation costs, scalability, lower quality of service and lack of real-time solutions, among others (El-Sayed et al., 2018).

In an EC architecture (Fig. 2), the end devices are not only data consumers, but also data producers. To minimize communication bottlenecks and latency, and to enhance performance, many of the operations are performed by nearby edge devices, such as base stations, access points or routers. Some tasks performed by edge devices are processing tasks on data that was offloaded, data caching and/or storage when long-term data storage is not needed and can be disposed of. In addition, using edge devices to perform computation will also improve security and privacy protection. Cloud data centers are used when long term storage or high computation resources are needed.

In this paper, the EC term will cover the previously mentioned architectures of Fog Computing, Cloudlets, Multi-access Edge Computing, and Mobile Cloud Computing.

Computation offloading is one of the capabilities of these systems and has been explored by several authors. If an application involves minor data communication but major computational processing, then computation offloading is the appropriate solution, which aspires to prolong battery lifetime (Taleb et al., 2017), benefiting from external computing resources and storage capacity (Li et al., 2018a). However, offloading also introduces costs as it implies higher systems and communication complexities.

Table 1 presents the challenges of computation offloading methods identified by different authors and highlights current approaches/solutions. In the following subsections, we will address in detail the analysis according to the main challenge tackled by the authors: energy consumption or battery saving, offloading complexity, and a group of other challenges.

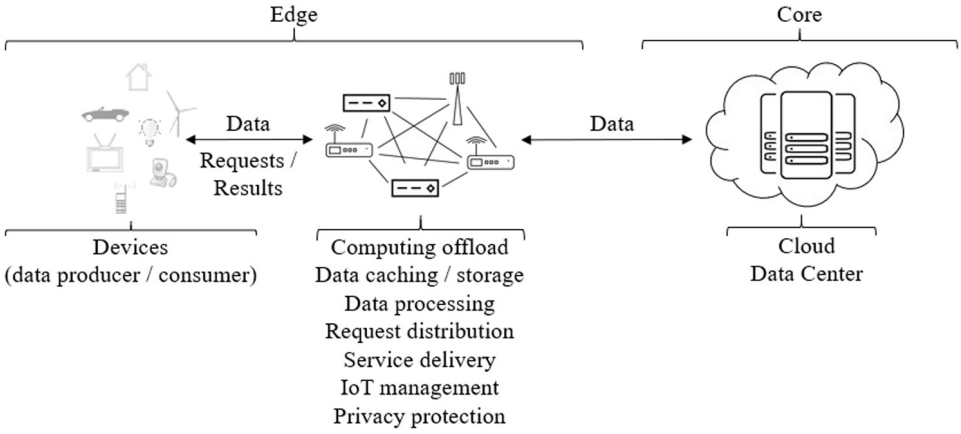

Fig. 2. Edge computing architecture.

3.1. Energy consumption

The following reviewed papers aim to address the challenge of **energy consumption or saving the devices' battery**.

Cuervo et al. (2010) presented *MAUI* which approaches code offloading in a Mobile Cloud Computing environment, regarding both energy and performance issues. *MAUI* analyzes the performance of the last processed method to decide dynamically whether offloading is beneficial.

Barbarossa et al. (2014) used an approach based on *FemtoClouds* which leverage the nearby unused end devices at the network edge, thus diminishing network latency while offloading computation to the traditional Cloud data center.

Chen et al. (2016) tackled the computation offloading decision-making problem as a *game theory* approach among multiple mobile device users for Mobile Edge-Cloud Computing.

Ur Rehman et al. (2016) presented a data mining offloading scheme. It is an opportunistic evaluation of the size of unprocessed data, privacy configurations, contextual information, and available on-board local resources, such as memory, CPU, and battery power. The offloading decision is performed on the end device. If it is beneficial and the edge resources are available, then the data is transmitted.

Wang et al. (2016) formulated a *nonconvex* problem to address minimization of both energy consumption by the smart device and execution latency. The proposed setup does not benefit from full offloading. Also, through analysis of the optimal solutions, the algorithm is able to successfully decide when to compute data locally.

Chen and Hao (2018) managed to achieve optimal task offloading in Software Defined Ultra-Dense Networks. By solving a NP-hard program the task offloading scheme was able to diminish the task duration while contemplating the user device energy consumption.

Wang et al. (2019) first approached the CPU clock frequency in the offloading decision. To minimize energy cost and queue delay, the authors proposed a distributed algorithm regarding clock frequency configuration, transmission power allocation, channel rate scheduling and offloading strategy selection. This algorithm optimizes both clock frequency for local execution and the transmission power in edge Cloud execution.

3.2. Offloading complexity

In this group the authors tackled challenges associated with **offloading complexity**, such as heterogeneity, availability and scalability.

Chun et al. (2011) introduced *CloneCloud* which uses a static program analyzer followed by dynamic program profiling to determine the partitioning of an application. These processes are done offline, then offloaded to the Cloud which runs a clone of the end device.

Kosta et al. (2012) proposed *ThinkAir*, a framework for code offloading which runs Mobile Cloud Computing applications on the Cloud through parallel computation.

Kemp et al. (2012) developed *Cuckoo* for Android and is a programming model for code offloading focused on the intermittent network connection.

Oueis et al. (2015) implemented an algorithm designed for cluster formation and load balancing. This cluster would be beneficial to the offloading process by joining resources, without increasing the system complexity.

Orsini et al. (2016) presented *CloudAware* to monitor networks, network strength, available surrogate computing resources for offloading, and the workload of each server, which was the responsibility of a discovery service.

3.3. Other challenges

Memory replication and **offloading quality** are the remaining challenges, that did not fit any of the previous categories.

Abdelwahab et al. (2016) introduced *REPLISOM*, aiming to trim Cloud responsiveness when multiple Internet of Things (IoT) devices replicate memory objects to the edge. The architecture enhances the Cloud Computing resources at the edge that provide clone virtual machines, storage and network resources for IoT applications.

Zhang et al. (2017) created a vehicular edge computing offloading scheme, where the several vehicular edge servers had a common backup server to share information. This process was addressed as a *game theory* through incentive mechanisms. Also, a distributed algorithm was implemented to reach the optimal computation offloading.

Computation offloading assumes multiple actions. Determining the cost of the offloading procedure needs to address some decisions, such as how to maximize the benefits of offloading; what will be the offloading target, code or data; what is the destination of the offloading process; will offloading consider the full task or just a minor part of it; and when should the offloading process occur. The decision between local processing or offloading is the outcome of all these choices. Latency, energy, and performance metrics are the main drivers of the process. Within this process, we can review Fig. 1 and zoom in on one of the steps in the flow of the offloading decision named "*Does the application benefit from offloading?*". Fig. 3 depicts the main questions the offloading decision must answer in order to achieve the proposed benefits. The questions presented aim to help the process of deciding whether the offloading will be beneficial or not.

How? The decision must sustain how does the application benefit from offloading, is through the improvement of the overall performance, by maximizing computation, or by reducing the energy consumption?

Table 1
EC approaches to computation offloading.

Subsection	Challenges	Approach	Reference
Energy consumption	Higher energy consumption when the latency to the server is low. Using profiling to estimate the energy savings of code offloading	Framework: MAUI Figuring if the method can and should be offloaded. Calculating the cost of offloading.	Cuervo et al. (2010)
	Optimizing the communication and computation resources jointly	Latency and energy constraints. Optimal resource allocation approach.	Barbarossa et al. (2014)
	Achieve efficient wireless access coordination Low energy efficiency and long data transmission time	Use a *game theory* algorithm to choose between local and Cloud computing, while achieving efficiency.	Chen et al. (2016)
	Limitations in computational power, battery resources, and memory availability	The decision algorithm considers data size, privacy settings, contextual information, and resource availability. The cost–benefit calculations are performed in the end device.	Ur Rehman et al. (2016)
	Energy consumption and latency minimization	Approach partial computation offloading by the optimization of computational speed of the smartphone, transmit power, and offloading ratio.	Wang et al. (2016)
	Minimizing latency, delay and devices' energy consumption	Addressed task placement and resource allocation for offloading optimization.	Chen and Hao (2018)
	Minimizing energy cost considering time and energy	Consider application completion deadline and processing capability constraints to choose the offloading destination.	Wang et al. (2019)
Offloading complexity	Heterogeneity in decisions	Framework: CloneCloud Not just yes/no decision but on the amount of offloaded applications. Removing the developer of the equation by making partitioning automatic and seamless. Calculating the cost of offloading.	Chun et al. (2011)
	Scalability	Framework: ThinkAir Perform on-demand resource allocation. Manipulates parallelism by dynamically creating, resuming, and destroying Virtual Machines.	Kosta et al. (2012)
	Complex programs for offloading	Framework: Cuckoo Custom runtime system. Resource manager. Programming model for developers.	Kemp et al. (2012)
	Customizable design Reduce complexity	Split the resources allocation process. Computation clusters for unsatisfied requests.	Oueis et al. (2015)
	Uninterrupted availability Limited requirements	Framework: CloudAware Holistic approach. Context adaptation. Guiding offloading decisions. Techniques to fit mobile applications and developer's mindset.	Orsini et al. (2016)
Other challenges	Memory replication	Framework: Replisom Splitting the Cloud resources to local resources. Compressed sampling construction.	Abdelwahab et al. (2016)
	Maintain offloading quality with limited resources	Used a *game theory* algorithm in vehicular edge, to design an optimal multilevel offloading scheme. Maximizes the utilities of both the vehicles and the computing servers.	Zhang et al. (2017)

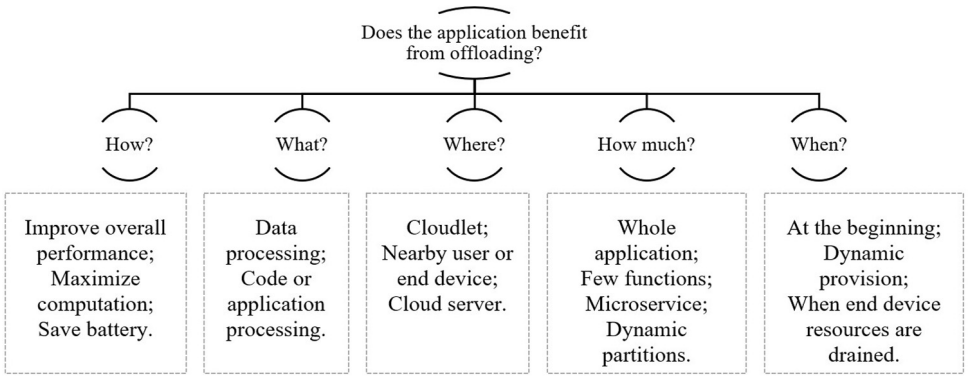

Fig. 3. Questions to decide if the offloading is beneficial.

What? What type of operations benefit from offloading? It is essential to determine if the offloading process will target data storage or code/application processing.

Where? It is crucial to identify resources for offloading so that the application benefit from the process. Will the process stay on the edge of the network, in a cloudlet, in a fog node, or in another end user device? Or, in alternative, be offloaded to the Cloud server on the core of the network. Such decision must also take into account the user mobility.

How much? Determining how much offloading will be performed is decisive. There is a need to decide the amount of data to transfer, or how many parts of the computation complexity to offload, while considering the networks' trade-offs.

When? It is critical to establish the ideal moment to offload a task or data. Among the alternatives are: at the beginning; only when the devices' resources cannot cope with the processing requirements; or with a dynamic provision, depending on the devices' resources availability or network conditions.

It is mandatory to know if the application benefits from offloading. It is only after this conclusion that the system will assess if offloading the application will improve computation or save energy. The cost of offloading is based on solving the challenges raised by each question. Accurately answering these five questions will enable the correct offloading decision and reduce latency, promoting users' quality of service and experience, thus the offloading decision will ultimately maximize utilization while saving energy.

The provided examples, share a common handicap, in terms of management of the computation offloading. These approaches are not adaptable to the environment and its changes, not learning from them. In the fast-changing environment we live in, they will quickly become obsolete, and thus are unacceptable solutions. Also, these approaches are only suitable for each cited scenario and are not widely-spreadable. To overcome these problems, AI arises as a promise for better EC computation offloading, contributing to making such approaches more transferable, adaptable, and scalable, increasing the systems or frameworks applicability to different scenarios, and not just one. In the next section, we will discuss the current AI approaches in EC systems, examining results, and pointing out limitations.

4. Artificial intelligence approaches to computation offloading in edge computing

In the previous section, we identified several cases of tasks and computation offloading applications to edge devices, none of which used AI techniques. Recently, several authors implemented AI in offloading with encouraging results, since it enhances the infrastructures' adaptability to continuously changing requirements, and reorganize themselves when necessary (Kiss et al., 2018). Bierzynski et al. (2017) stated that increased efforts to apply AI techniques on the edge of the network are required to unlock its full potential, advancing the developments of supporting hardware and communication infrastructures. Edge Intelligence is a term already used by some authors, such as Deng et al. (2019) and Zhou et al. (2019), as the integration between EC and AI. Zhou et al. (2019) discussed the lack of collaboration among edge and Cloud, as one limitation of the Edge Intelligence scope, direct at deploying AI models on the edge servers or devices.

AI has its origins in the middle of the past century. McCarthy (1987) considered AI as the science of creating intelligent machines, including intelligent computer programs, and ML as conceiving systems that can adapt and learn within an environment. ML is a subfield of AI that supplies computers the knowledge to learn without being explicitly programmed (Samuel, 1959).

Depending on the feedback available to the learning system, we can classify ML tasks into Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Reinforcement Learning (Ayodele, 2010). The first two are the most common and explored, however, Reinforcement Learning use is increasing.

Supervised Learning is the most frequent type of ML. This type of learning consists of the data and the predicted output for every combination of inputs. The aim is to map the relationship between the input variables and its outputs so that, for a new set of inputs, it will predict the correct output. The learning process results in the term *supervised*, where the algorithm iteratively seeks to learn to predict from the dataset provided and is corrected until it achieves a predefined error rate.

Unsupervised Learning has no such supervision, meaning that for a combination of inputs, there is no class or label for it. Here the goal is to find a structure or distribution in the input space so that the algorithms can find recurring patterns.

Semi-supervised Learning is located in between supervised and unsupervised learning, as it has both labeled and unlabeled data. Because labeling data is very time expensive, usually only a small set of data is labeled. Semi-supervised learning may refer to either *transductive* or *inductive learning*, where transduction means to label the unlabeled data through an approximate model, and the latter to the prediction function for the input data.

Reinforcement Learning interacts with the environment while learning a model that outlines situations to actions, as to maximize the reward. The algorithms observe the produced results in each interaction. The model must find the best actions to take to achieve the best reward because it is not stated which direction is the best. Each iteration impacts the next iteration that will encompass the previous reward. Trial-and-error characterizes this type of learning. Reinforcement learning does not learn from data, it learns from its own experience.

In this section, we provide a literature review of applications of AI techniques in computation offloading through edge devices, organized in families of ML algorithms by similarity. We observed the following classes of algorithms in the EC offloading literature: Neural Networks and Deep Learning, Regression, a single group of Instance-based, Association Rule Learning, Decision Tree and Bayesian. We compiled a set of other ML algorithms, that do not fit the previously mentioned categories, in the *Other Algorithms* subsection. Furthermore, we provide an in-depth analysis of the use of AI for offloading, in particular the use case of Vehicular Edge Computing Networks. Finally, we provide a summary of all encountered algorithms, together with a brief analysis.

4.1. Neural networks and deep learning algorithms

An artificial Neural Network is a model of computation inspired in the structure of neural networks in the brain. It can be described as a directed graph, where the nodes are the neurons and the edges are the links between them. Each neuron has as input a weighted sum of the outputs of the neurons connected to its incoming edges (Shalev-Shwartz and Ben-David, 2013). However, these are limited, due to the training time and model complexity amongst others, and deep neural networks, currently called deep learning, emerged to surpass traditional approaches in training neural networks. These are very specific neural networks, with more players, different protocols, and more complexity. Nowadays, these reach great performance on several problems such as computer vision, speech recognition, and natural language processing (Nielsen, 2015). Deep learning adopts numerous layers to implement computational models and embody data abstractions. Having its roots in conventional neural networks, recent studies state that it considerably outperforms its predecessors (Shyu et al., 2018). Both of these algorithms belong to the Supervised Learning style.

Yu et al. (2017) developed a multi-label classification approach to coordinate offloading, and to minimize the computation and offloading overhead they established a Deep Supervised Learning method in mobile edge computing devices. Their results show that the proposed framework can decrease system cost up to 49.24%, 23.87%,

15.69%, and 11.18% compared to the following schemes: "no offloading", "random offloading", "total offloading" and "multi-label linear classifier-based offloading", respectively.

Kang et al. (2017) used a systematic approach to identify the optimal points to partition computation and designed *Neurosurgeon*, a scheduler to autonomously segregate Deep Neural Network computation between end devices and data centers. *Neurosurgeon* is highly adaptable and using its intelligence partitions computation for latency or mobile energy enhancements. *Neurosurgeon* was evaluated through a mobile development platform, achieving an average improvement in latency by 3.1x, up to 40.7x, energy consumption diminished on average by 59.5%, up to 94.7%, and data center throughput attained increases averaging 1.5x, up to 6.7x.

Alelaiwi (2019) addressed the difficulties of determining the best node to receive the offloaded computation in an edge/fog–Cloud environment. By proposing a Deep Learning based approach, they can select the node with the shortest response time, after learning the request and response-time history of the nodes and predicting the response time of the future request. The performance evaluation of the prediction models between the two fog nodes and a Cloud node resorted to the Mean Absolute Percentage Error (MAPE) and R-squared, which were normalized, and all three models achieve a MAPE under 0.1 and a R-squared greater than 0.6, meaning high accuracy in response time prediction, thus the node with minimized response time was selected.

The works that made use of Deep Learning Algorithms achieved promising results when compared to baseline solutions, especially the works of Yu et al. (2017), Kang et al. (2017) and Alelaiwi (2019) that provided quantitative results and allow a better understanding of how much better their approach was. Furthermore, the specific algorithms presented were Deep Supervised Algorithm to calculate offloading accuracy and Deep Neural Network to calculate latency, energy consumption, and data throughput, respectively. Despite using different techniques to evaluate contrasting features, both achieved good results indicating a wide range of applications for these algorithms.

In addition, to configure and manage networks intelligently and autonomously, Deep Learning algorithms are feasible (Fadlullah et al., 2017). Moreover, these algorithms automatically extract features minimizing human effort and domain knowledge to gather discriminating characteristics (Shyu et al., 2018), making them highly appreciated in the heterogeneity of EC environments. Furthermore, these techniques may be applied in different scenarios, despite having been developed for specific use cases, meaning that both methodology and architecture are generalizable.

4.2. Regression algorithms

Regression problems are a form of supervised learning. Regression tries to predict continuous values, based on the relationship between two variables, and how much one influences the other. These methods attempt to explicitly model the relationship between the inputs and the outputs. The model is refined, after each iteration, based on the measure of the error in the predictions made. In the go-to methodology the algorithm builds a model on the features of training data and uses this model to predict values for new data. Some examples of applications are financial forecasting, trend analysis, marketing, time series prediction, and even drug response modeling (Alpaydin, 2014). In this subsection, we will review the literature that uses regression algorithms to perform computation offloading. These are Supervised algorithms.

Khoda et al. (2016) motivated by cost-effective offloading decisions, developed an intelligent code offloading decision-making system. *ExTrade* improves application response time while saving end devices' energy. Using a nonlinear optimization solution, the offloading decision is made by the Lagrange Multiplier which inputs intelligence to the system to capture its environment and achieve high accuracy in offloading prediction, save computation time and energy. To estimate execution time, the authors used a Statistical Regression-based algorithm, which considered the dynamic behavior of the environment and application usage.

Kwon et al. (2016) proposed *f_Mantis*, a feature-based forecasting technique to overcome the input-sensitivity challenge of mobile application performance. Regarding the metrics of execution time, energy consumption, memory usage, and state size, the system calculates if running the application is more profitable than offloading it. Through a Non-Linear Regression algorithm, *f_Mantis* predicts both dividends and costs of offloading a method somewhere in the application execution, contributing to factual offloading decisions that diminish running time or energy waste of a mobile application.

Khoda et al. (2016) only considered one Cloud service provider for a mobile device in the 5G system, in the system evaluation and when compared to *ThinkAir*, *ExTrade* over-performed considering computation time saving, prediction accuracy, and energy-saving metrics. Regarding the *f_Mantis* evaluation (Kwon et al., 2016), the system achieved high rates of accuracy when forecasting the gains and costs of offloading, granting precise offloading decisions that allowed saving energy and trimming running time of a mobile application.

The results regarding Regression Algorithms are also promising as a technique to coordinate computing offloading in EC systems. The amount of published works is lower than the one identified in the previous category of ML algorithms, as well as the use cases since the authors do not provide its feasibility in real-world scenarios, instead, they provide theoretical problem solving through regression algorithms.

4.3. Instance-based, association rule learning, decision trees and Bayesian algorithms

Instance-based are non-parametric methods, also called memory-based learning, that stores the training instances in a lookup table and interpolate from these. An Association Rule is an implication of form X -> Y, where X is the antecedent and Y is the consequent of the rule, i.e., Y is the dependent variable. These methods are used in data mining applications, e.g. web sites that recommend books, movies, music, among others (Alpaydin, 2014). A Decision Tree, also a non-parametric method, is a hierarchical data structure implementing the divide-and-conquer strategy. The local region is identified in a sequence of recursive splits in a smaller number of steps (Alpaydin, 2014). Bayesian rules are used to calculate the conditional probabilities of the classes having a direct influence on each other (Alpaydin, 2014). All of these are Supervised Learning algorithms, except for Association Rule that belongs to the Unsupervised Learning style.

Eom et al. (2013) evaluated the possibility to use ML techniques in an adaptive scheduling problem, regarding the memory limitations, in a mobile offloading framework. They used the classifiers of RandomTree (Decision Trees Algorithms), Instance-Based Learning (Instance-based Algorithm), and Rule-Based Learning (Association Rule Learning Algorithms) and compared the three in the scheduling problem. In the provided evaluation, Instance-Based Learning was the best, operating 7% better than RandomTree and 3% than Rule-Based Learning. Regarding the best fit, Instance-Based Learning, they proceed to prove its benefits and the scheduler adaptability to dynamic network circumstances. Using an image processing workload to be offloaded under changeable network bandwidth conditions, they managed to make the right scheduling decision in 87.5% of the occurrences.

Eom et al. (2015) developed *MALMOS* (MAchine Learning-based Mobile Offloading Scheduler) which is a ML classifier that decides whether mobile computations are run locally rather than offloaded while implementing a training mechanism. This feature permits the scheduling to dynamically accommodate its decision under previous offloading verdicts and their precision. The authors evaluated performance and cost of three ML algorithms: Instance-Based Learning (Instance-Based Learning Algorithms), Perceptron (Artificial Neural Networks Algorithms), and Naïve Bayes (Bayesian Algorithms), regarding the classifier training time, classification time, and scheduling

accuracy. *MALMOS* adaptability was tested regarding several network circumstances and computing potential of external resources using an Android-based prototype. Under its offloading accuracy evaluation, *MALMOS* outperformed two scheduling policies, threshold-based and linear equation-based, by 10.9% to 40.5%.

Crutcher et al. (2017) aimed to diminish resource consumption through the identification of the correct edge node to collect an offloaded task. By integrating elements from Knowledge-Defined Networking and historical data, they managed to introduce smart predictions on offloading costs. The nodes for computation offloading were selected through a k-Nearest Neighbor (Instance-based Algorithm). Their results exposed the modeling accuracy with regression over numerous network metrics, and in certain cases, k-Nearest Neighbor queries using Euclidean distance are more promising rather than rectilinear distance.

Junior et al. (2019) addressed the problem of offloading underachievement when ignoring contextual information, incurring in imprecise decisions that result in the deficient performance of the overall system. To achieve a great accuracy level in the offloading decision, the authors contemplated a Context-Sensitive Offloading System that clouts on ML reasoning techniques and powerful system profiling. For the decision-making on whether the computation should be offloaded or not, this work used k-Nearest Neighbors (Instance-based Algorithm), Rules (Association Rule Learning Algorithms), Decision Tree (Decision Trees Algorithms) and Naive Bayes (Bayesian Algorithms). Moreover, two different evaluations took place. The first evaluated the best two performing algorithms, J48 (C4.5) and JRIP (Rules-based), within their database of classification algorithms which had the previous two-plus IBK (k-Nearest Neighbors) and Naive Bayes algorithms, and achieved a 95% offloading decision accuracy. The second, contemplated structured and authentic scenarios, with shifts on context information between experiments, which attained precise decisions while enhancing performance and energy consumption.

Overall, all the mentioned algorithms achieved auspicious results. Systems were compared in terms of scheduling accuracy in Eom et al. (2015), energy consumption and time sending data in Crutcher et al. (2017), in Junior et al. (2019) specificity, sensitivity, precision, F1, false positive rate and false negative rate, and, finally, in Eom et al. (2013) bandwidth and offloading decision accuracy. All approaches outperformed the baseline systems or in the case of the work developed in Eom et al. (2013) the authors compared amongst each other three classifiers in an offline offloading scheduler. Nevertheless, these approaches fail to use real-world scenarios and are limited to theoretical approaches to the quality of their algorithms. Except for the work developed in Junior et al. (2019), the only that test a real scenario, face recognition, using a Cloudlet device to receive the offloaded application.

4.4. Other algorithms

In this subsection, we will detail different algorithms that do not fit the previous categories. Some of these are well studied and documented, such as the Markov Decision Process and Q-Learning. Markov models use a parametric random process to generate the input sequences, and since the states are not observable, whenever the model visits a state, an observation is recorded that is a probabilistic function of the state. Q-learning searches the optimal policy of offloading by a trial-and-error method, thus it is essential that the decision-maker calculates the trade-off between exploration and exploitation in an unknown environment (Cao et al., 2019). Markov models and Q-learning, aim to learn the best policy, which is the sequence of actions that maximize the total reward (Alpaydin, 2014). Most of the algorithms presented in this category belong to the Reinforcement Learning style. Nevertheless, others, such as Genetic approaches are Supervised.

Zhang et al. (2015) developed an optimal offloading algorithm for handling connectivity fluctuations in Cloudlets. Both the users' mobility and local load, as well as both Cloudlets' admission control and availability, were considered to analytically calculate the success probability of effective offloading schedules. By defining and determining a Markov decision process model aiming to diminish computation and offloading costs, authors achieved an optimal policy for the end user, which attained great results when compared to conventional baseline schemes.

Alam et al. (2016) proposed a Reinforcement Learning multi-agent-based code offloading coordination in a fog environment, delivering to end users a low-latency service. End devices are heterogeneous and for that reason, resource requirements change over time and space becoming one major challenge. Their evaluation, made by simulations, mainly studied energy consumption and response time, and achieved good results compared to only phone or Cloud approaches and compared with the *ThinkAir* system, which is a framework with both dynamic adaptation and scaling of computational performance, developed for computation offloading in Mobile Cloud Computing environments (Kosta et al., 2012).

Cao and Cai (2018) investigated the challenge of making a computation offloading decision in a multi-user environment. Regarding a Cloudlet-based Mobile Cloud Computing system, they developed a non-cooperative game which contemplates communication and computation costs of computation offloading. This game includes at least one pure-strategy Nash Equilibrium Point, which means no player benefits from altering only their own strategy, thus the introduction of ML to customize each end device's decision-making regarding the unknown and changing environment. Authors developed a Fully Distributed Computation Offloading Algorithm to encounter the Nash Equilibrium Point where no data sharing is allowed. Their algorithm augmented the number of Cloudlets computing end devices' data, keeping a low execution cost across the system when compared to local computation by end devices.

Chen et al. (2018) used a Multi-access Edge Computing environment considering the high availability of base stations for computation offloading, which was modeled as a Markov decision process. The goal was to enhance performance, regarding the task and energy queues states, and the channel qualities between end users and base stations, in the offloading decision. They introduced a double deep Q-network, named *DARLING*, to determine the best computation offloading policy in a changeable network. Furthermore, a Q-function decomposition technique was merged with the double deep Q-network, named *Deep-SARL*, resulting in a new learning algorithm to solve stochastic computation offloading. The evaluation determined considerable gains in computation offloading performance compared with the baseline policies.

Wang et al. (2018a) merged Deep Reinforcement Learning techniques and Federated Learning framework with the mobile edge systems, revising caching and communication. They intended the *In-Edge AI* framework to intelligently use the cooperation between devices and edge nodes. Aiming to cut avoidable communication, the framework shared the learning criteria, so the models could be provided with better training, inference and carried dynamic system-level and application-level improvements. Edge caching and computation offloading in mobile edge systems were the scenarios under evaluation, in which the *In-Edge AI* framework had great performance results.

Tan et al. (2018) developed a Deep Reinforcement Learning framework for vehicular networks, aiming to access an excellent resource allocation of communication, caching and computing. In addition, to reach operational and cost supremacy the vehicle's mobility was explored. Their theoretical evaluation presented substantial gains in cost efficiency.

Li et al. (2018b) framed the offloading and resource allocation for Mobile Edge Computing system as a Deep Reinforcement Learning optimization problem, with the goal to trim the sum cost through processing delay and energy consumption of all users in this system. The scheme was evaluated using simulations, with several systems characteristics, such as data size, computational capacity, and a number of

user's equipment, registering surpassing performance when compared to other baseline solutions.

Alam et al. (2019) applied a deep Q-Learning based method to automatically manage the offloading requests. The decision regards the resource demand and availability, and the network status to minimize latency. They used a Markov decision process to allocate computing resources and reinforcement learning in their solution. The evaluation of the model was based on simulations of the N-queen problem, and regarding response time and energy consumption analysis. They compared their results with *ThinkAir* and achieved better results both in response time and energy consumption.

Qiu et al. (2019) proposed a new model-free Deep Reinforcement Learning-based online computation offloading approach, which they called deep reinforcement learning combined with a genetic algorithm (DRGO), for blockchain-empowered Mobile Edge Computing. They implemented a Markov decision process for blockchain tasks, such as mining and data processing. In addition, to maximize long-term offloading performance, they used Deep Reinforcement Learning to adapt to very dynamical environments. The authors also addressed the computational complexity, adopting an adaptive genetic algorithm, in order to accelerate the convergence that results from high-dimensional action space. Moreover, for the online offloading problem, the authors implemented a Markov decision process to consider dynamic environments. The evaluation was performed under a simulation environment with the following metrics: average cost, task drop rate, and average transmission time. Also, they compared their solution to other three representative benchmark policies, such as greedy, genetic and deep deterministic policy gradient algorithms. The results showed that the DRGO algorithm achieved better performance compared to the mentioned benchmarks.

Cheng et al. (2019) presented an innovative approach to computation offloading in an edge environment considering remote energy and computation constraints. They presented a space-air-ground integrated network (SAGIN) edge/Cloud computing architecture composed of Unmanned Aerial Vehicles (UAVs) as edge serves with virtual machines for tasks offloading and satellites for Cloud computing. The authors used a Markov decision process for the offloading decision, where the system state considers the network dynamics. A Deep Reinforcement Learning-based approach is used to improve the decision policy in large action space scenarios for its applicability in the decision-making process. The authors used simulation to evaluate the weighted sum of delay, energy consumption, and server usage cost. The results exhibit that the proposed edge virtual machine allocation and task scheduling approach can achieve near-optimal performance with very low complexity and the proposed learning-based computing offloading algorithm not only converges fast but also achieves a lower total cost compared with other offloading approaches, in this case, a 'Greedy' and a 'Random' approach.

Although the mentioned works do not provide quantitative evaluations, the stated results mention an overall better performance than baseline schemes. The metrics used to evaluate each work were the following: number of beneficial Cloudlet Computing end devices in Cao and Cai (2018), Cloudlet availability and user's relative expected cost in Zhang et al. (2015), and trade-off among the computation task execution delay, the task drops, the task queuing delay, the Multi-access Edge Computing service payment and the task failure penalty in Chen et al. (2018). All these approaches are dedicated to measuring the performance of offloading algorithms in more theoretical scenarios.

4.5. Offloading in Vehicular Edge Computing Networks

In the previous subsections, we provided a global review of offloading using AI algorithms. Because those works use different approaches, address distinct areas, and have divergent opportunities, created by diverse datasets, algorithms, and testbeds, we think that an in-depth analysis of a particular use case would help us in better understanding the current state-of-the-art on the use of AI for offloading. We chose this use case because the involved technology reached prominent relevance with the development of autonomous, semi-autonomous, and assisted driving. Moreover, we decided to leave an analysis of approaches without AI algorithms out of the scope of this paper, mainly because the authors were more concerned with the vehicular network architecture, rather than the offloading decision.

With the concentration and tremendous growth in the number of computers, vehicular infrastructure, communication capabilities, and automobiles technologies, researchers in vehicular networks have perceived new opportunities to prosper (Ahmed and Gharavi, 2018). The research community acknowledges this as one technology to solve a list of traffic-related issues, such as traffic congestion, traffic accidents, and environmental pollution (Wu et al., 2020). This evolution also poses a prominent challenge because the requirements of data communication, computation, and storage are higher than the currently available (Hou et al., 2016). Motivated by this progress, researchers investigated several possibilities for computation and data offloading between vehicles and other vehicles (V2V), vehicles and infrastructures (V2I), and between vehicles and everything (V2X).

Vehicular Cloud networks have contributed to better resource utilization and higher computation performance. However, the connection delay may depreciate the offloading efficiency and increase the network operation cost (Liu et al., 2018). Hence, the use of EC devices, such as Base Stations (BS) or Road Side Units (RSU), have promising applications and results. These can significantly diminish communication latency, which scales exponentially with the increase of routing hops in vehicular networks (Ning et al., 2019a). Besides, the high number of parked vehicles in urban areas presents itself as a valuable opportunity by putting these underutilized vehicular resources into use (Hou et al., 2016).

In these environments, the offloading decision is remarkably important. According to Ning et al. (2019b) such a decision should include the metrics of the *utility function*, such as the server reward for message uploading, message size, offloading cost, and delay. However, these metrics may not be enough to provide an intelligent offloading system. For this reason Ning et al. (2019a) and Liu et al. (2018) identify the following challenges:

- How to manage extremely dynamic vehicular networks?
- How to build an accurate AI model in this fast changing environment?
- How to migrate these AI model implementation models?
- How to determine computation offloading strategies to maximize the quality of experience?
- How to handle intermittent connectivity between vehicles and EC servers?
- How to implement adequate pricing schemes to advocate resource sharing of vehicles?

Fig. 4 illustrates the different opportunities for offloading in a vehicular network. It performs computation offloading on three scenarios. Vehicles to other vehicles (V2V), either moving or parked, where available resources support the computation of the offloaded tasks. Vehicles communicate with RSUs (V2R), which can perform computation and produce forwarding decisions locally. Finally, vehicles may connect directly with a BS (V2I), which are powerful devices to perform local computation. Both RSUs and BSs frequently use a wired connection to the Cloud servers for long-term data storage or, in case of great necessity, use Cloud resources also for computation.

Feng et al. (2017) implemented a distributed vehicular architecture, where users inside vehicles can perform computation offloading to nearby nodes, such as RSUs, based on their distributed decisions. The authors developed a computation offloading algorithm based on Ant Colony Optimization (ACO), aiming to maximize the *utility function* of offloaded tasks concerning the delay, tasks caching, and the algorithm's

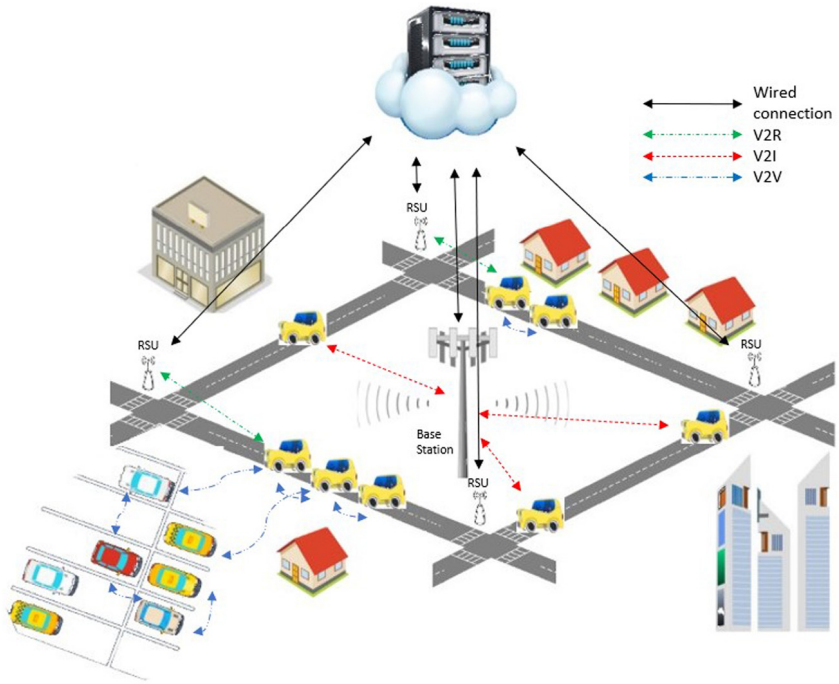

Fig. 4. Vehicular Edge Computing Network implementation.

efficiency to manage them. The authors, through simulations in urban and highway scenarios, compared three offloading scenarios: local processing, Naive algorithm, and ACO. The last achieved the best results due to higher flexibility in the scheduling algorithm.

Wang et al. (2018c) stated that an efficient offloading in vehicular networks is difficult to achieve due to the delay requirements and variable computation resources. To address these issues, the authors developed a Vehicular Fog Computing System (VFCS) with a priority queuing system. Also, a Semi-Markov Decision Process (SMDP) was applied in an application-aware offloading policy, to maximize the reward of the VFCS, by satisfying the delay requirements, and achieve the optimal computation resource allocation, through dynamic resource assignment. The authors used a simulation environment with three offloading approaches, a First Come First Served, a Preemptive (higher-priority tasks are processed first), and their SMDP proposal. SMDP performed better compared to the other two.

Sun et al. (2018) used a modified genetic algorithm to tackle computation offloading in edge clouds and vehicular clouds. The authors subdivide the offloaded computation into smaller tasks in order to optimize the response time. The main issues mentioned were the mobility of the vehicles and the different processing capabilities of each. The genetic-based heuristic algorithms were implemented to reduce the complexity of the mentioned issues while improving the use of vehicular onboard resources. Through simulations, the authors analyzed several metrics, such as the time each vehicle remained in the vehicular cloud, the offloading decisions, the average response time and the system stability. Without any comparison with other implementations, it was stated that the solution improved the overall computing efficiency of the edge cloud.

Qi et al. (2019) implemented a deep reinforcement learning offloading decision model, which takes into consideration the required resources, the access network, and the vehicles' mobility. Based on the gathered offloading knowledge, the decision for the present task reflects the future data dependency of the subsequent tasks. The simulations performed show that the authors' approach converged on the offloading decision, faster and with more adaptability to different conditions than greedy algorithms.

Cui et al. (2019) presented a multi-platform intelligent offloading and resource allocation algorithm. This algorithm can dynamically manage the computing resources to enhance performance. The platform calculates the destination of the computation offloading by a K-nearest neighbor, and the authors tackle the optimization problem of resource allocation through reinforcement learning. The authors assessed the proposed joint optimization algorithm, that manages the local, BS/RSUs, and Cloud resources, through simulation. The algorithm succeeded to decrease latency cost, and reduce the average system cost by 50%, and 80%, as opposed to full local computation and full BS/RSU offloading, respectively.

Ning et al. (2019a) presented their work on deep reinforcement learning to develop an intelligent offloading system for Vehicular Edge Computing. The authors applied a Dynamic V2I Matching algorithm (DVIM), to match the vehicles and BS scheduling. Also, they developed finite Markov chains to model the communication and computation states and a Mobility-Aware Double DQN (MADD) to assess vehicles' mobility. The joint optimization problem, between task scheduling and resource allocation strategy, aimed to maximize users' quality of experience. The authors tested, through simulations, DVIM against a greedy method, and a random sorting method, where DVIM achieved a good overall performance and decreased execution time by over 90%. And MADD surpassed traditional Deep Q-Network (DQN), Q-learning, and two baseline algorithms, from 10% to 15%.

When a moving vehicle that is processing a task leaves the range of the system, it breaks the offloading process and diminishes the overall performance. This challenge was tackled by Wu et al. (2020), and the authors implemented an optimal offloading scheme that considers the departure of vehicles engaged with offloaded tasks in a Vehicular Fog. The authors elaborated on the offloading problem as a semi-Markov decision process, which had to determine and analyze the states, actions, discounted rewards, and transition probabilities. The authors also designed the algorithm to find the optimal task offloading scheme, while maximizing the long-term reward of the vehicular system. Under experimental scenarios, compared to a greedy scheme, the suggested algorithm scored higher system rewards.

Sun et al. (2020) developed a Joint Offloading Decision and Task Scheduling algorithm (*JODTS*), a hybrid intelligent optimization algorithm based on a partheno-genetic algorithm (Hou et al., 2012) and heuristic rules for computation offloading in multi-user and multi-server Vehicular Edge Computing scenarios. Aiming to optimize overall

accuracy of the decision, task delay and computing resource consumption, associated with reducing the time complexity, the offloading algorithm not only regulates task execution place but also specifies the execution order of the tasks on the server. In a comprehensive evaluation through simulations, *JODTS* outperformed four baseline algorithms, achieving better offloading utility.

Ning et al. (2020) presented a Distributed Joint Offloading Algorithm (*DJOA*), applying intelligence in an offloading cost minimization problem while diminishing latency for 5G-enabled vehicular networks. The authors tackled the problem by dividing it into two subproblems: centralized unlicensed spectrum allocation and distributed cellular spectrum allocation. To address the first, the authors developed a two-sided matching algorithm to allocate the unlicensed spectrum. For the second, the authors implemented a deep reinforcement learning method to schedule cellular channels. Also, the authors simplified the system state to achieve a distributed offloading scheduling, which could decrease the communication overhead between vehicles and the macrocell. The authors simulated a real-world scenario to assess *DJOA* against DQN, Q-learning and full cellular offloading algorithms, and achieved an overall better offloading cost.

Finally, Chen et al. (2020) used a vehicle-to-vehicle (V2V) communication to perform computation offloading between vehicles concentrated in traffic lights or other areas, thus fully exploring the idle assets. The authors planned the computation offloading process as a Min–Max problem between one task and various cooperative vehicles, aiming to diminish latency while considering mobility as part of the problem. Also, to minimize the task execution time, the authors implemented the Max–Min Fairness algorithm for the decision to call all service vehicles, and a Particle Swarm Optimization algorithm to meet the fittest task allocation scheme. The authors produced several simulations to assess their model, outperforming other schemes in time execution and consumed resources.

What these works showed us is that although some problems have been solved with great success, some remain open, such as **resource discovery** by fast-moving vehicles that need to find instantly nearby devices to perform offloading and data storage (Datta et al., 2016); **assessing the computational capacity** of nearby devices, to weigh on the offloading decision, and the need for mobility models, discovering critical vehicular behaviors (Hou et al., 2016); **enhancing offloading efficiently** in the presence of various applications with diverse delay requirements (Wang et al., 2018b); **guaranteeing constant wireless connectivity** between vehicles and RSUs, for a continuous offloading process (Premsankar et al., 2018); **achieving the full benefits of deploying deep reinforcement learning** algorithms in this dynamic and agile environment (Ning et al., 2019a).

4.6. Summary

We summarize in chronological order, within each group of algorithms, the literature review considering the analysis on which EC paradigm was addressed, the ML algorithm used, the motivation for their work, the adopted metrics, and the results obtained by each of the cited works. In Table 2 we address the three groups of Neural Networks and Deep Learning Algorithms, Regression Algorithms, and Instance-based, Association Rule Learning, Decision Trees and Bayesian Algorithms, each group separated by a double line. In Table 3 we present all the authors in the "Other Algorithms" group. Finally, Table 4 summarizes computation offloading for only Vehicular Edge Computing Network.

The analyzed EC paradigms include Multi-access Edge Computing, which also encompasses Base Stations (BS) and Road Side Units (RSU), Fog Computing, Edge Computing, and Mobile Cloud Computing. The ML algorithms (MLA) used were Deep Supervised Learning (DSL), Deep Neural Network (DNN), Reinforcement Learning (RL), Deep Reinforcement Learning (DRL), Convolutional Neural Networks (CNN), Statistical Regression-based Algorithm (SRA), Non-linear Regression (NlR), Instance-based Algorithms (IbA), Artificial Neural Networks (ANN), Bayesian, Association Rule Learning (ARL), Decision Tree (DT), Fully Distributed Computation Offloading Algorithm (FDCOA), Markov decision process (Markov), Deep Reinforcement learning combined with Genetic algOrithm (DRGO), Ant Colony Optimization (ACO), Modified Genetic Algorithm (MGA), Integer Linear Program (ILP), Partheno-Genetic Algorithm and Heuristic Rules (PGA/HR), and Particle Swarm Optimization (PSO).

The surveyed works are not directly comparable. Use cases are not the same, different metrics and datasets were also used to assess unlike solutions, and the results are mainly qualitative. Thus, we cannot conclude which of the previous algorithms is the best. Without further implementations, under comparable conditions, with the same dataset, running in analogous hardware, with the same metrics, we cannot provide a direct comparison between the results each author achieved in their work. Furthermore, the surveyed systems lack real-life implementation, thus the evaluations performed, which achieved promising results, may accomplish very different outcomes in realistic scenarios on computation offloading decisions. Moreover, the dataset used in each proposed algorithm is absent or shortfalls from an explicit definition of the data source and how it was trained. Some approaches are compared to different benchmarks or systems that have a high rate of acceptance among the research community, however, others only use performance-related metrics to evaluate their solution. In addition, some solutions fail to address the multi-resources availability and heterogeneity of the real-world and were enforced to single node offloading. Also, these implemented solutions analyzed specific metrics, without the desirable generalization to the possibility of incorporating other metrics. Finally, these works address a specific section of the offloading process, resulting in the need to further explore other algorithms or offloading schemes to solve the existing problems in each phase of the process.

Table 5 indicates the type of information provided by the authors in their work, from which we highlight the available datasets ("-" no available dataset, "✓" available either directly or indirectly through references, and "★" if the dataset is not available, but the provided information can be used to train a model); the design and description of the algorithms, despite the varying level of detail presented; and the testbed used, comprising information on the hardware and technologies employed in the evaluation of their work. The purpose is to provide information regarding reproducible experiments and simulations. Moreover, most of the works do not provide a public dataset to reuse. However, several authors presented a comprehensive discussion around the algorithm and how it was used, which could, eventually, be reproduced.

Amongst the discussed works, there are some promising results, such as the ones presented in the works of Yu et al. (2017) and Kang et al. (2017), when using deep learning algorithms. Also, Li et al. (2018b) achieved interesting results when using deep reinforcement learning. In addition, the techniques used by Yu et al. (2017) may also be applied to multi-user, multi-cell scenarios, and mobility management. And Li et al. (2018b) developed techniques to deal with interference and radio resources. Moreover, concerning the discussed use case, the works of Cui et al. (2019) that used an instance-based algorithm, and Ning et al. (2019a) which provided a deep reinforcement learning algorithm, displayed numerical results, allowing to assess the effectiveness of the authors' proposals.

Yu et al. (2017) proposed a deep learning scheme that reduces system cost by half and achieved an offloading accuracy around 60%. Kang et al. (2017) performed a comprehensive evaluation using eight deep neural networks datasets to benchmark across different types of wireless networks, also against the framework MAUI. *Neurosurgeon* used a deep neural network and achieved lower latency (between 3.1x and 40.7x), savings in energy consumption (from 59.5% to 94.7%), and higher throughput (1.5x up to 6.7x). Li et al. (2018b) used a *sum cost* of the key elements of reinforcement learning: *state*, *action*, and *reward*, achieving an overall improvement around 150%.

Table 2
Summary of AI techniques in computation offloading in EC.

Reference	EC	Algorithm	Motivation	Metrics	Results
Yu et al. (2017)	MEC	DSL	Minimize the offloading cost considering the network resource usage	Offloading accuracy	Reduces system cost up to 49.24%, 23.87%, 15.69%, and 11.18% compared to "no offloading", "random offloading", "total offloading" and "multi-label linear classifier-based offloading", respectively
Kang et al. (2017)	MEC	DNN	Processing and computation partitioning strategies	Latency, energy consumption and data throughput	Achieves an improvement in latency from 3.1x, up to 40.7x, energy consumption diminished from 59.5%, up to 94.7%, and data center throughput attained increases from 1.5x, up to 6.7x
Alelaiwi (2019)	FG	DNN	Identifying the best node for offloading	Response time	Achieves a MAPE under 0.1 and a R-squared greater than 0.6
Khoda et al. (2016)	MCC	SRA	Cost-effective offloading decision	Execution time, energy consumption, prediction accuracy and time saving	*ExTrade* over performed in the metrics under evaluation
Kwon et al. (2016)	MCC	NlR	Predict performance	Offloading decision predictions accuracy, overhead and energy consumption	High rates of accuracy when forecasting the gains and costs of offloading, granting precise offloading decisions which saves energy and reduces running time of a mobile application
Eom et al. (2013)	MCC	IbA ARL DT	Adaptative scheduling	Bandwidth and decision accuracy	The Instance-Based Learning was the best, 7% better than RandomTree and 3% better than Rule-Based Learning
Eom et al. (2015)	MCC	IbA ANN Bayesian	Scheduling	Scheduling accuracy	*MALMOS* out performed two scheduling policies, threshold-based and linear equation-based, from 10.9% to 40.5%
Crutcher et al. (2017)	MEC	IbA	Identifying which edge nodes should receive tasks	Energy consumption and time sending data	Several network metrics can be modeled accurately
Junior et al. (2019)	MCC	IbA ARL DT Bayesian	Underperformance when the offloading decision ignores contextual information	Specificity, sensitivity, precision, F1, FPR and FNR, energy, time and context	Guarantees performance gains and energy efficiency

Table 3
Summary of AI techniques in computation offloading in EC ("Other Algorithms").

Reference	EC	Algorithm	Motivation	Metrics	Results
Zhang et al. (2015)	MCC	Markov	Intermittently connected Cloudlet system	Cloudlet availability and user's relative expected cost	Achieves an optimal policy for the end user, which attained great results in cost reduction
Alam et al. (2016)	FG	RL	Low-latency service delivery	Energy consumption and the response time	Good results compared to only phone or Cloud approaches and compared with *ThinkAir*
Cao and Cai (2018)	MCC	FDCOA	Offloading decision making	Number of beneficial Cloudlets computing enddevices	Increases the number of Cloudlet Computing end devices' data, keeping a low execution cost across the system
Chen et al. (2018)	MEC	Markov	Maximize utility performance	Trade-off among the computation execution delay, task drops, task queuing delay, MEC service payment and the failure penalty	Considerable gains in computation offloading performance
Wang et al. (2018a)	MEC	DRL	Collaboration between devices and edge nodes	Performance between utility and training time	Proved to achieve near-optimal performance
Tan et al. (2018)	EC	DRL	Resource allocation of communication, computing and caching	Performance/cost	Registered surpassing performance
Li et al. (2018b)	MCC	DRL	Privacy	Running time, energy consumption and memory cost	Improvement of 150% in object recognition when compared to a generic deep model
Alam et al. (2019)	EC	RL Markov	Automatic Offloading Management	Response time and energy consumption	Outperformed *ThinkAir* in the metrics under evaluation
Qiu et al. (2019)	MEC	DRGO	Blockchain mining and processing data	Average cost, Task drop rate Average transmit time	Better performance compared to three benchmarks
Cheng et al. (2019)	EC	Markov DRL	Energy and computation constraints	Weighted sum of delay, energy consumption, server usage cost	Near-optimal performance with very low complexity; Validates the convergence and efficiency of proposed approaches

Table 4
Summary of AI techniques in computation offloading in Vehicular Edge Computing Network.

Reference	EC	Algorithm	Motivation	Metrics	Results
Feng et al. (2017)	RSU	ACO	Maximize the sum utility of offloaded tasks	Scheduling decisions accuracy	Achieves best results due to higher flexibility in the scheduling algorithm, compared to local processing, Naive algorithm, and Ant Colony Optimization
Wang et al. (2018c)	Fog	Markov	Maximize reward of delay and resource allocation	Offloading policy	Achieves best performance compared First Come First Served, and Preemptive approaches
Sun et al. (2018)	BS	MGA	Vehicles' mobility; Different processing capabilities	Contact duration, offloading decisions response time and the system stability	Improved overall computing efficiency of the edge Cloud
Qi et al. (2019)	BS	DRL	Offloading complexity	Learning capacity	The algorithm converged on the offloading decision, faster and with more adaptability to different conditions than greedy algorithms
Cui et al. (2019)	BS/RSU	IbA	Dynamically manage the computing resources for offloading	Sum cost of bandwidth, CPU frequency, and vehicles distance	Decreases latency cost, and reduces the average system cost by 50%, and 80%, as opposed to full local computation and full BS/RSU offloading
Ning et al. (2019a)	BS/RSU	DRL	Maximize users' quality of experience	Task scheduling and resource allocation	Good trade-off between network performance and execution time by over 90% compared to greedy and random sorting methods. Surpassed traditional DQN, Q-learning, and two baseline algorithms, from 10% to 15%.
Wu et al. (2020)	RSU	Markov	Finding the optimal offloading scheme also considering the departure of vehicles	States, actions, discounted rewards, and transition probabilities	Compared to a greedy scheme, the suggested algorithm scored higher system rewards
Sun et al. (2020)	BS/RSU	PGA/HR	Optimize task delay and computing resource consumption	Accuracy, delay cost, operating cost, and offloading utility	*JODTS* outperformed four baseline algorithms, achieving better offloading utility
Ning et al. (2020)	BS/RSU	DRL	Minimize cost and latency	Offloading scheduling, communication overhead	Assessed *DJOA* against DQN, Q-learning and full cellular offloading algorithms, and achieved an overall better offloading cost
Chen et al. (2020)	RSU	PSO	Minimize latency and task execution time	Execution time under different situations	Outperforming other schemes in time execution and consumed resources

Cui et al. (2019) through a K-nearest neighbor achieved a latency and system cost reduction by 50%, and 80%, respectively, thus successfully managing the computing resources for offloading. Ning et al. (2019a) developed a Matching algorithm to match the vehicles and BS scheduling, and Markov chains to model the communication and computation states and to assess vehicles' mobility, achieving a good overall performance and a decreasing execution time by over 90%.

These highlighted works presented promising quantitative results. Notwithstanding the need to assess them using the same datasets, metrics and testbeds, for a direct comparison between these and other mentioned works.

Regarding the offloading decision, only seven authors addressed this crucial issue. Khoda et al. (2016) motivated by cost-effective offloading decisions, used a regression algorithm to achieve high accuracy in offloading predictions, save computation time and energy. Cao and Cai (2018) contemplated communication and computation costs on the computation offloading decision. Sun et al. (2018) studied mutually dependent tasks, because of the influence inter-dependency of tasks held on the offloading decision. The authors analyzed the scheduling time slot of individual tasks and the processing vehicles for any scheduling slot. Junior et al. (2019) address the problem of offloading underachievement when ignoring contextual information, the authors implemented an several algorithms and achieved a 95% offloading decision accuracy, while enhancing performance and energy consumption. Qi et al. (2019) also investigated the offloading decision as a resource scheduling problem and considered the decision as a long-term planning problem. Cheng et al. (2019) used a Markov decision process for the offloading decision, where the system state considers the network dynamics. Sun et al. (2020) settled the decision to offloading based on the most recently learned information regarding the vehicles' task offloading strategies and the service nodes' task scheduling policies.

5. Discussion

In this section, we will discuss, and highlight some important ideas on computation offloading, such as the environment where it takes place, the methods used to achieve a successful state in the running application and, finally, the advantages and limitations of AI methods in EC, considering training and deployment.

The first idea is on the environment, meaning that resource-restricted devices may have the possibility for computation offloading to nearby edge servers or, further away, Cloud servers. As mentioned before, the devices' restrictions on battery and computation resources, especially IoT devices and end-users' end devices, such as smartphones, may require computation offloading to the Cloud. The centralized Cloud Computing has exceptional computing resources available, receiving the offloading of computation-demanding applications, which may diminish the computation delay and the devices' battery usage. However, the shortcomings of Cloud Computing, already mentioned in Section 3, may fail to achieve the delay requirements of sensitive applications, because the span between users and the Cloud servers results in long transmission delays. Thus, using any EC architecture, with computing resources near the end user, delivers efficient and flexible computing services (Cheng et al., 2019). Moreover, deploying ML algorithms in the Edge (either in nodes or servers), even if highly

Table 5
Available datasets or reproduceable testbeds.

Reference	Available dataset	Algorithm design	Testbed
Yu et al. (2017)	–	✓	✓
Kang et al. (2017)	⋆	–	✓
Alelaiwi (2019)	–	–	✓
Khoda et al. (2016)	⋆	✓	✓
Kwon et al. (2016)	⋆	✓	✓
Eom et al. (2013)	✓	–	✓
Eom et al. (2015)	–	✓	✓
Crutcher et al. (2017)	✓	✓	✓
Junior et al. (2019)	⋆	✓	✓
Zhang et al. (2015)	⋆	✓	✓
Alam et al. (2016)	⋆	✓	✓
Cao and Cai (2018)	–	✓	–
Chen et al. (2018)	⋆	✓	✓
Wang et al. (2018a)	–	✓	–
Tan et al. (2018)	⋆	✓	✓
Li et al. (2018b)	–	✓	✓
Alam et al. (2019)	⋆	✓	✓
Qiu et al. (2019)	⋆	✓	✓
Cheng et al. (2019)	⋆	✓	✓
Offloading in Vehicular Edge Computing Networks			
Feng et al. (2017)	⋆	✓	✓
Wang et al. (2018c)	⋆	✓	–
Sun et al. (2018)	⋆	✓	✓
Qi et al. (2019)	⋆	✓	✓
Cui et al. (2019)	⋆	✓	✓
Ning et al. (2019a)	⋆	✓	✓
Wu et al. (2020)	⋆	✓	✓
Sun et al. (2020)	⋆	✓	✓
Ning et al. (2020)	⋆	✓	✓
Chen et al. (2020)	⋆	✓	✓

– no available data.
✓ available either directly or indirectly through references.
⋆ no available data, but the provided information can be used to train a model.

resource-demanding, is becoming feasible because of the improvements of edge resources and capabilities. These new hardware capacities may cause to consider the Edge as the future main target for ML applications.

Next, we will provide a discussion on the use of AI, more precisely ML algorithms, to improve the offloading decision and the following steps after.

To begin with, traditional approaches, without any AI method to support the decision, require previous knowledge regarding end users' patterns and network parameters. While also needing the same previous knowledge, AI methods adapt better to new circumstances without human interaction. In a static or slowly varying environment, the traditional approach may be beneficial and even recommended. Besides, usually developing solutions within the traditional approach consider *ad hoc* optimization, meaning it may fail to engage the long-term performance of computation offloading or apply in additional scenarios. Furthermore, traditional approaches are not able to tackle, at least so efficiently, some network problems in offloading processes, such as instability, heterogeneity, and the inter-dependency of computing tasks (Sun et al., 2018). The overall environment where these processes occur is stochastic and identified by fast-changing mobile networks, where there is no prior knowledge, or where knowledge is quite limited to support the decision-making process. This characteristic leads to the term *Concept Drift*, meaning the capacity to learn in dynamic environments (Žliobait, 2010), where data used to train and test the model do not efficiently apply. Thus, IA, specifically ML, enables the system to learn patterns from new distinct and varying data streams (Iwashita and Papa, 2019).

The value of ML-based approaches arises from the implemented mechanisms to detect changes or new patterns of the ever-changing environment (Cao et al., 2019). Also, ML-based implementations regulate, effectively, the varying workloads in the users' devices from the offloading process, such as local processing, new requests, and ended executions. The network features and computing capacities of edge servers change throughout the timeline of the application offloading. The number of connected devices also change, and accurate decisions must be instant, which AI aims to achieve with no human interaction. ML algorithms can detect data patterns that would be hard to find without these systems, being able to predict future situations. ML solutions should apply in several scenarios and consider the environments' evolution to accomplish durable high performance of computation offloading. The slow convergence caused by high-dimensional action space still is an open issue (Qiu et al., 2019).

Different ML algorithms cope differently with these changing environments. Re-training Neural Networks could be feasible, but the entire network had to sustain this process, which would be very time consuming, even more with Deep Learning. In these dynamic scenarios, processing an application locally without ML can be advantageous, because of re-training times, or deterioration in predictive performance. Working with an obsolete model that does not incorporate new data into the training data, thus ignores current information regarding this changing behavior, enhances the difficulty of an accurate offloading decision. It is remarkably important to address *concept drift* so that the ML model does not lose its predictive performance. Žliobait (2010) and Brownlee (2016) mention some solutions, from which we extracted the most relevant ones for this work, *Periodically Re-Fit*, *Periodically Update* and *Learn the Change*. *Periodically Re-Fit* is the method to update the static model, from time to time, with more current data. *Periodically Update*, uses the present state as the origin for a fit process that updates the model fitness, applying a sample of the most recent data, rather than reject the static model. Finally, *Learn the Change*, is a solution where a new model learns to improve the predictions from the static model based on the relationships in more up-to-date data.

Finally, we address the advantages and limitations of training the models of the previously presented algorithms.

5.1. Advantages

In this section, we summarize the advantages of some of the algorithms, based on the works of Atkeson et al. (1997), Alpaydin (2014), and Cao et al. (2019).

Neural Networks and Deep learning algorithms have good performance in predicting stochastic changes for offloading decision-making optimization. Another advantage is that, compared to other algorithms, Neural Networks do not require so much feature engineering. Nevertheless, these still need to have defined parameters to continue its evolution, since going too slow or too fast may lead to not achieve the best solution. Neural Networks perform parameter tuning to converge on the best solution between the compromise of quality and speed. This manual setup will allow the Neural Network to learn from data without any additional intervention.

Regression algorithms are ideal for continuous values, for example, to calculate the expected execution time of a given application in order to decide afterward if offloading computation would be beneficial.

Instance-based algorithms are considered to cover the diverse and variable requirements of EC, through classification and cluster methods to create offloading decisions for varied service features. Moreover, estimating or predicting radio parameters and anomaly detection in wireless networks are possible applications. Also, these algorithms do not suffer from data interference, thus receiving inputs about an operating regime does not diminish modeling performance, resulting in a feasible approach for offloading decision-making.

Association Rule algorithms are good at discovering interesting relations between variables in large databases. Also, at predicting future behaviors using past data. These advantages result in the applicability of the algorithms in the offloading decision based on previous experiences.

Decision Trees algorithms allow a fast localization of the region covering an input. Characterized as a binary decision, comparable to the offloading compromise (as we explained in Section 1), each decision discards the other branches' possibilities, thus significantly reducing the size of the dataset. Because these are highly interpretable, with understandable *If-Then* rules, these are very popular and sometimes preferred over more accurate but less interpretable methods. Also, Decision Trees are used more frequently for classification than for regression. These algorithms also learn and respond quickly.

Bayesian algorithms assume that each of the features is conditionally independent of one another, require less training data and are highly scalable. Moreover, these algorithms are used to make probabilistic future predictions. Thus, the offloading process may benefit from predictions about the network state and the available resources, since these algorithms can handle continuous and discrete data.

Markov decision process algorithms are applied to model the decision to whether or not perform offloading in a continuous process. This process consists of moving amongst states and calculating the corresponding reward. Only if the calculated reward is positive, other steps are addressed, such as choosing the appropriate EC server, and the amount of workload to offload.

Q-Learning algorithms are appropriate for decision-making of the offloading process when the available information is restricted and the environment is dynamic. Also, to design the online and model-free learning approach, without any prior knowledge of the environment.

5.2. Limitations

Although ML algorithms are, in general, beneficial to the offloading decision problem, these algorithms also suffer some limitations. Thus, before implementing them, we must also be aware of their limitations.

Neural Networks and Deep Learning have a long training time, long response delay, and mostly rely on massive labeled data (Cao et al., 2019). In fast-changing environments, this would have a great cost associated with the offloading decision, thus, when deploying these algorithms, developers must take the scenario into consideration. Furthermore, it is mainly subjective, knowing how a prediction is accomplished because it is remarkably challenging to perceive it from the Neural Network model. Deep Learning algorithms having a higher number of layers, usually hidden, emphasizes this condition. Being *black-box* systems, understanding its holistic functioning, troubleshooting, and optimization is remarkably hard.

Some **Regression** algorithms have dimensionality limitations and do not allow interactions of predictor variables. Besides, when a dependent variable is not normally distributed, Regression algorithms only apply in case of a large sample size. Also, they have the choice of a loss, meaning that there is a one-sided decision that can be made (Smola and Vishwanathan, 2009). Regression algorithms may not manipulate irrelevant features well, especially if the features are strongly correlated. Meaning that the algorithms will not discard these irrelevant data, contributing to the growth of the size of the dataset. Because the offloading decision is addressed as a categorization process, as we explained in Section 1, Regression algorithms may display some adversities, since they are optimal for continuous values.

Instance-based algorithms rely on massive amounts of data that involve a possibly large amount of memory to store, and it is hard to display their theoretical performance bounds (Cao et al., 2019). As these algorithms are very sensitive to data, each request for different information involves starting the identification of a new local model from scratch, thus rising the implementation time (Atkeson et al., 1997). With a scenario with user mobility, where each offloading decision involves the assessment of a different set of conditions (that must originate a new request), the use of this algorithm may require a constant overhead.

Association Rules algorithms have a limitation with small samples, because even if there is a dependency with a strong confidence value, if the number of such samples is small, the association rule between variables is worthless (Alpaydin, 2014), increasing the risk of an incorrect offloading decision.

Decision Trees algorithms are usually slower than other approaches, overfitting becomes an issue if a high purity tree is desirable and prone to outliers. Also, the tree may grow to be very complex while training complicated datasets and loses valuable information while handling continuous variables (Atkeson et al., 1997). In the EC agile environment, where time is a great concern and the offloading decision must not only be accurate but also fast, using Decision Trees algorithms may not always be feasible.

Bayesian algorithms are not sensitive to irrelevant features, resulting in large datasets because these algorithms keep all data for re-training the model, not discarding the irrelevant values. Bayesian algorithms have extreme difficulty in specifying a *prior* model, because there is a necessity to specify every setting of the model parameters, to define a *posterior*, which is a probability distribution that represents the updated model. This difficulty leads to an increase in computational complexity. Besides, prediction consumes more memory compared to others. These limitations, in resource-restricted devices, may put at risk the correct offloading decision.

The algorithms under the "Others" category, can learn without *a priori* knowledge. However, they comprise dimensionality limitations, imply a trade-off between exploration and exploitation and rely on hand-crafted features. If we extend the analysis to deep reinforcement learning, a long training time in large discrete state space and extreme computational complexity are added to the shortcomings of these algorithms (Cao et al., 2019). This trade-off and hand-crafted features may lead to an incorrect offloading decision. Reinforcement learning needs a high amount of data and computation. Also too much reinforcement learning may produce an overload of states, thus diminishing the results. Finally, reinforcement learning wrongly assumes the world is Markovian, in which future events have a probability of happening according to the state attained in the previous event.

6. Conclusions and future work

In this paper, we have surveyed computation offloading within EC environments. First, we assess computation offloading in EC systems and how the different authors addressed the challenges that computation offloading still poses. Next, we reviewed the application of AI techniques, more precisely ML algorithms, for computation offloading regarding EC paradigms. We identified the category of each algorithm, the metrics used in each study and the results achieved. We also provided an in-depth analysis of the use of AI for offloading, in particular, the use case of Vehicular Edge Computing Networks, where we presented an overview, the state-of-the-art approaches using AI algorithms, and open issues. Finally, we discussed the usage of AI methods in EC environments.

We provided an indirect comparison between the referenced works because the authors used different ML techniques or algorithms, and addressed multiple features and metrics to quantify their work on the use of ML algorithms to coordinate computation offloading. From the performed analysis and provided discussion, we highlight the following works. The method developed in Junior et al. (2019) achieved the higher accuracy value in the offloading decision of whether performing it or not through the use of a Markov decision process. Using a deep learning algorithm, more precisely a deep neural network, the authors in Kang et al. (2017) attained impressive results in terms of latency, energy consumption, and data throughput, with improvements up to 40.7x, 94.7%, and 6.7x, respectively. Within the discussed use case, through a K-nearest neighbor, the work of Cui et al. (2019) accomplished a latency and system cost reduction by 50%, and 80%. By developing a Matching algorithm, the author in Ning et al. (2019a) achieved a good performance and diminished execution time by over 90%.

We highlighted the main advantages and limitations of ML algorithms used to assist the offloading decision, and concluded that Deep Learning and Markov decision process algorithms can achieve significant performance and accuracy for offloading decision coordination.

Computation offloading is beneficial merely if processing an application locally uses more time and/or energy than the offloading overhead (Akherfi et al., 2018). As we discussed, there are many considerations to be made in the offloading decision, which drastically increase in complexity in EC environments. Applying ML algorithms to sustain this decision shows auspicious results. A correct offloading decision is not trivial to achieve, and the associated complexity is a perfect scenario for implementing ML-based approaches. The promising results supported by AI, which consider past experiences, and have fast and powerful models, improve the decision-making process, and the accuracy of a correct offloading decision.

As future work, we plan to test some aforementioned algorithms, in identical setups, using the same technologies, hardware, and under the same environment, to compare the results and assess which is better in providing a correct offloading decision. We intend to develop a framework that suits all stages of the offloading process, instead of single-phase solutions and study the results of switching the type of criteria used in each offloading decision. Finally, the objective is to implement a system that can automatically and transparently decide amongst these several criteria accordingly to the environment on which the offloading may occur.

CRediT authorship contribution statement

Gonçalo Carvalho: Conceptualization, Investigation, Writing - original draft, Writing - review & editing, Visualization. **Bruno Cabral:** Conceptualization, Writing - review & editing, Visualization, Supervision. **Vasco Pereira:** Writing - review & editing, Visualization, Supervision. **Jorge Bernardino:** Writing - review & editing, Visualization, Supervision.

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.

Acknowledgments

This work is supported by the MobiWise project: from mobile sensing to mobility advising (P2020 SAICTPAC/0011/2015), co-financed by COMPETE 2020, Portugal 2020-POCI, European Regional Development Fund of European Union, and the Portuguese Foundation of Science and Technology.

We also acknowledge the support from the European Regional Development Fund (FEDER), through the Regional Operational Programme of Lisbon (POR LISBOA 2020) and the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework [Project 5G with Nr. 024539 (POCI-01-0247-FEDER-024539)].

References

Aazam, M., Zeadally, S., Harras, K.A., 2018. Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities. Future Gener. Comput. Syst. 87, 278–289.

Abdelwahab, S., Hamdaoui, B., Guizani, M., Znati, T., 2016. Replisom: Disciplined Tiny Memory Replication for Massive IoT Devices in LTE edge cloud. IEEE Internet Things J. 3, 327–338.

Ahmed, A., Ahmed, E., 2016. A survey on mobile edge computing. In: 10th International Conference on Intelligent Systems and Control (ISCO). pp. 1–8.

Ahmed, E., Gharavi, H., 2018. Cooperative vehicular networking: A survey. IEEE Trans. Intell. Transp. Syst. 19, 996–1014.

Akherfi, K., Gerndt, M., Harroud, H., 2018. Mobile cloud computing for computation offloading: Issues and challenges. Appl. Comput. Inform. 14, 1–16.

Alam, G., Hassan, M., Uddin, Z., Almogren, A., Fortino, G., 2019. Autonomic computation offloading in mobile edge for IoT applications. Future Gener. Comput. Syst. 90, 149–157.

Alam, M.G.R., Tun, Y.K., Hong, C.S., 2016. Multi-agent and reinforcement learning based code offloading in mobile fog. In: International Conference on Information Networking. pp. 285–290.

Alelaiwi, A., 2019. An efficient method of computation offloading in an edge cloud platform. J. Parallel Distrib. Comput. 127, 58–64.

Alpaydin, E., 2014. Introduction to Machine Learning, third ed. The MIT Press.

Atkeson, C.G., Moore, A.W., Schaal, S., 1997. Locally weighted learning. Artif. Intell. Rev. 11, 11–73.

Ayodele, T.O., 2010. Types of machine learning algorithms. In: New Advances in Machine Learning. pp. 19–48.

Barbarossa, S., Sardellitti, S., Di Lorenzo, P., 2014. Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks. IEEE Signal Process. Mag. 31, 45–55.

Bierzynski, K., Escobar, A., Eberl, M., 2017. Cloud, Fog and Edge: Cooperation for the Future?. In: 2nd International Conference on Fog and Mobile Edge Computing (FMEC) Cloud. pp. 62–67.

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. pp. 13–16.

Brownlee, J., 2016. Master machine learning algorithms: Discover how they work and implement them from scratch. In: Machine Learning Mastery.

Cao, H., Cai, J., 2018. Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach. IEEE Trans. Veh. Technol. 67, 752–764.

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, 56–62.

Chen, C., Chen, L., Liu, L., He, S., Yuan, X., Lan, D., Chen, Z., 2020. Delay-optimized V2V-based computation Offloading in Urban Vehicular edge computing and networks. IEEE Access 8, 18863–18873.

Chen, M., Hao, Y., 2018. Task offloading for mobile edge computing in software defined ultra-dense network. IEEE J. Sel. Areas Commun. 36, 587–597.

Chen, X., Jiao, L., Li, W., Fu, X., 2016. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Netw. 24, 2795–2808.

Chen, J., Ran, X., 2019. Deep learning with edge computing: A review. Proc. IEEE 107, 1655–1674.

Chen, X., Zhang, H., Wu, C., Mao, S., Ji, Y., Bennis, M., 2018. Optimized Computation Offloading Performance in Virtual Edge Computing Systems Via Deep Reinforcement Learning. IEEE, pp. 1–31.

Cheng, X., Lyu, F., Quan, W., Zhou, C., He, H., Shi, W., Shen, X., 2019. Space/Aerial-assisted computing offloading for IoT applications: A learning-based approach. IEEE J. Sel. Areas Commun. 37, 1117–1129.

Chun, B.G., Ihm, S., Maniatis, P., Naik, M., Patti, A., 2011. CloneCloud: Elastic execution between mobile device and cloud. In: ACM EuroSys'11. pp. 181–194.

Crutcher, A., Koch, C., Coleman, K., Patman, J., Esposito, F., Calyam, P., 2017. Hyperprofile-Based Computation Offloading for Mobile Edge Networks.

Cuervo, E., Balasubramanian, A., Cho, D.k., Wolman, A., Saroiu, S., Chandra, R., Bahl, P., 2010. MAUI: Making Smartphones last longer with code Offload. In: 8th International ACM Conference on Mobile Systems, Applications, and Services, (MobiSys'10). pp. 49–62.

Cui, Y., Liang, Y., Wang, R., 2019. Resource allocation algorithm with multi-platform intelligent offloading in D2D-enabled vehicular networks. IEEE Access 7, 21246–21253.

Dastjerdi, A., Gupta, H., Calheiros, R., Ghosh, S., Buyya, R., 2016. Chapter 4 – Fog Computing: principles, architectures, and applications. In: Internet of Things. pp. 61–75.

Datta, S.K., Da Costa, R.P.F., Harri, J., Bonnet, C., 2016. Integrating connected vehicles in Internet of Things ecosystems: Challenges and solutions. In: 17th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'16). pp. 1–6.

Deng, S., Member, S., Zhao, H., Yin, J., Dustdar, S., 2019. Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence.

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, 1587–1611.

El-Sayed, H., Sankar, S., Prasad, M., Puthal, D., 2018. Edge of Things: The big picture on the integration of edge, IoT and the cloud. IEEE Access 6, 1706–1717.

Enzai, N.I.M., Tang, M., 2014. A taxonomy of computation offloading in Mobile Cloud Computing. In: 2nd IEEE Int. Conf. Mob. Cloud Comput. Serv. Eng.. IEEE, pp. 19–28.

Eom, H., Figueiredo, R., Cai, H., Zhang, Y., Huang, G., 2015. MALMOS: Machine learning-based mobile offloading scheduler with online training. In: 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering. pp. 52–60.

Eom, H., Juste, P.S., Figueiredo, R., Tickoo, O., Illikkal, R., Iyer, R., 2013. Machine learning-based runtime scheduler for mobile offloading framework. In: IEEE/ACM 6th International Conference on Utility and Cloud Computing, (UCC'13). IEEE, pp. 17–25.

Fadlullah, Z.M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T., Mizutani, K., 2017. State-of-the-art deep learning: Evolving Machine Intelligence Toward Tomorrow's intelligent network traffic control systems. IEEE Commun. Surv. Tutor. 19, 2432–2455.

Fei, X., Shah, N., Verba, N., Chao, K.M., Sanchez-Anguix, V., Lewandowski, J., James, A., Usman, Z., 2019. CPS data streams analytics based on machine learning for cloud and fog computing: A survey. Future Gener. Comput. Syst. 90, 435–450.

Feng, J., Liu, Z., Wu, C., Ji, Y., 2017. AVE: Autonomous vehicular edge computing framework with ACO-based scheduling. IEEE Trans. Veh. Technol. 66, 10660–10675.

Fernando, N., Loke, S.W., Rahayu, W., 2013. Mobile cloud computing: A survey. Future Gener. Comput. Syst. 29, 84–106.

Garcia Lopez, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., Riviere, E., 2015. Edge-centric computing. ACM SIGCOMM Comput. Commun. Rev. 45, 37–42.

Hall, P., Miller, H., 2018. Fog computing architecture, evaluation, and future research directions. IEEE Commun. Mag. 46–52.

Hou, X., Li, Y., Chen, M., Wu, D., Jin, D., Chen, S., 2016. Vehicular Fog Computing: A viewpoint of vehicles as the infrastructures. IEEE Trans. Veh. Technol. 65, 3860–3873.

Hou, Y., Li, L., Wang, L., 2012. Picking routes optimization of automated warehouse based on Parthenon genetic Ant Colony algorithm. In: International Conference on Convergence Information Technology (ICCIT'12). pp. 198–204.

Huang, D., Wang, P., Niyato, D., 2012. A dynamic offloading algorithm for mobile computing. IEEE Trans. Wireless Commun. 11, 1991–1995.

Iwashita, A.S., Papa, J.P., 2019. An overview on concept drift learning. IEEE Access 7, 1532–1547.

Junior, W., Oliveira, E., Santos, A., Dias, K., 2019. A context-sensitive offloading system using machine-learning classification algorithms for mobile cloud environment. Future Gener. Comput. Syst. 90, 503–520.

Kang, Y., Hauswald, J., Gao, C., Rovinski, A., Mudge, T., Mars, J., Tang, L., 2017. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In: Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS '17. pp. 615–629.

Kemp, R., Palmer, N., Kielmann, T., Bal, H., 2012. Cuckoo: A computation Offloading Framework for smartphones. In: Mobile Computing Application and Service. pp. 59–79.

Khan, A.U.R., Othman, M., Madani, S.A., Khan, S.U., 2014. A survey of mobile cloud computing application models. IEEE Commun. Surv. Tutor. 16, 393–413.

Khoda, M.E., Razzaque, A., Almogren, A., Mehedi, M., Atif, H., Alelaiwi, A., 2016. Efficient computation offloading decision in Mobile Cloud Computing over 5G Network. Mob. Netw. Appl. 21, 777–792.

Kiss, P., Reale, A., Ferrari, C.J., Istenes, Z., 2018. Deployment of IoT applications on 5G edge. In: IEEE International Conference on Future IoT Technologies, (Future IoT). pp. 1–9.

Kosta, S., Aucinas, A., Hui, P., Mortier, R., Zhang, X., 2012. ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: IEEE INFOCOM. pp. 945–953.

Kumar, K., Liu, J., Lu, Y.H., Bhargava, B., 2013. A survey of computation offloading for mobile systems. Mob. Netw. Appl. 18, 129–140.

Kwon, Y., Yi, H., Kwon, D., Yang, S., Cho, Y., Paek, Y., 2016. Precise execution offloading for applications with dynamic behavior in mobile cloud computing. Pervasive Mob. Comput. 27, 58–74.

La, H.J., Kim, S.D., 2014. A taxonomy of offloading in mobile cloud computing. In: IEEE 7th International Conference on Service-Oriented Computing and Applications, (SOCA'14). IEEE, pp. 147–153.

Li, J., Gao, H., Lv, T., Lu, Y., 2018b. Deep reinforcement learning based computation offloading and resource allocation for MEC. In: IEEE Wireless Communications and Networking Conference, WCNC. p. 6.

Li, C., Xue, Y., Wang, J., Zhang, W., Li, T., 2018a. Edge-oriented computing paradigms: A survey on architecture design and system management. ACM Comput. Surv. 51, 39:1—-39:34.

Liu, Q., Su, Z., Hui, Y., 2018. Computation Offloading Scheme to Improve QoE in Vehicular Networks with Mobile Edge computing. In: 10th International Conference on Wireless Communications and Signal Processing (WCSP'18). IEEE, pp. 1–5.

Liu, L., Zhang, Z.H.I., Zhang, P., 2019. Energy-aware mobile edge computation offloading for IoT over heterogenous networks. IEEE Access 7, 13092–13105.

Mach, P., Becvar, Z., 2017. Mobile edge computing: A Survey on Architecture and computation Offloading. IEEE Commun. Surv. Tutor. 19, 1628–1656.

McCarthy, J., 1987. Generality in artificial intelligence. Commun. ACM 30, 1030–1035.

Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R.H., Morrow, M.J., Polakos, P.A., 2018. A comprehensive Survey on Fog Computing: State-of-the-Art and research challenges. IEEE Commun. Surv. Tutor. 20, 416–464.

Nielsen, M.A., 2015. Neural networks and deep learning. http://dx.doi.org/10.1201/b22400-15.

Ning, Z., Dong, P., Wang, X., Rodrigues, J., Xia, F., 2019a. Deep reinforcement learning for vehicular edge computing: An intelligent offloading system. ACM Trans. Intell. Syst. Technol. (TIST) 10, 60:24.

Ning, Z., Li, Y., Dong, P., Wang, X., Obaidat, M.S., Hu, X., Guo, L., Guo, Y., Huang, J., Hu, B., 2020. When deep reinforcement learning meets 5G-enabled Vehicular Networks: A distributed Offloading Framework for traffic big data. IEEE Trans. Ind. Inf. 16, 1352–1361.

Ning, Z., Wang, X., Huang, J., 2019b. Mobile edge computing-enabled 5G Vehicular Networks: Toward the Integration of Communication and Computing. IEEE Veh. Technol. Mag. 14, 54–61.

Orsini, G., Bade, D., Lamersdorf, W., 2016. Computing at the mobile edge: Designing elastic android applications for computation offloading. In: 8th IFIP Wireless and Mobile Networking Conference, (WMNC'15). pp. 112–119.

Oueis, J., Strinati, E.C., Barbarossa, S., 2015. The fog balancing: Load distribution for small cell cloud computing. In: IEEE 81st Vehicular Technology Conference, (VTC'15). pp. 1–6.

Pang, Z., Sun, L., Wang, Z., Tian, E., Yang, S., 2016. A survey of Cloudlet based Mobile Computing. In: International Conference on Cloud Computing and Big Data, (CCBD'15). pp. 268–275.

Patel, M., Hu, Y., Hédé, P., Joubert, J., Thornton, C., Naughton, B., Julian, R.R., Chan, C., Young, V., Tan, S.J., Lynch, D., 2014. Mobile edge computing – Introductory technical White Paper. ETSI White Pap. 11, 1–36.

Premsankar, G., Ghaddar, B., Di Francesco, M., Verago, R., 2018. Efficient placement of edge computing devices for vehicular applications in smart cities. In: IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, (NOMS'18). IEEE, pp. 1–9.

Qi, Q., Wang, J., Ma, Z., Sun, H., Cao, Y., Zhang, L., Liao, J., 2019. Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach. IEEE Trans. Veh. Technol. 68, 4192–4203.

Qiu, X., Liu, L., Chen, W., Hong, Z., Zheng, Z., 2019. Online deep reinforcement learning for computation Offloading in Blockchain-Empowered mobile edge computing. IEEE Trans. Veh. Technol. 68, 8050–8062.

Rahimi, M.R., Ren, J., Liu, C.H., Vasilakos, A.V., Venkatasubramanian, N., 2014. Mobile cloud computing: A survey, state of art and future directions. Mob. Netw. Appl. 19, 133–143.

Roman, R., Lopez, J., Mambo, M., 2018. Mobile edge computing, Fog, et al.: A survey and analysis of security threats and challenges. Future Gener. Comput. Syst. 78, 680–698.

Samuel, A.L., 1959. Some studies in Machine Learning using the Game of Checkers. IBM J. Res. Dev. 3, 210–229.

Satyanarayanan, M., Bahl, P., Cáceres, R., Davies, N., 2009. The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. 8, 14–23.

Shalev-Shwartz, S., Ben-David, S., 2013. Understanding machine learning: From theory to algorithms.

Shyu, M.L., Chen, S.C., Iyengar, S.S., 2018. A survey on deep learning: Algorithms, techniques. ACM Comput. Surv. 51, 1–36.

Smola, A., Vishwanathan, S., 2009. Introduction to Machine Learning. University Press, Cambridge I, p. 213.

Sun, J., Gu, Q., Zheng, T., Dong, P., Valera, A., Qin, Y., 2020. Joint optimization of computation Offloading and Task Scheduling in Vehicular edge computing networks. IEEE Access 8, 10466–10477.

Sun, F., Hou, F., Cheng, N., Wang, M., Zhou, H., Gui, L., Shen, X., 2018. Cooperative Task Scheduling for computation Offloading in Vehicular Cloud. IEEE Trans. Veh. Technol. 67, 11049–11061.

Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., Sabella, D., 2017. On Multi-Access edge Computing: A survey of the emerging 5G Network edge cloud Architecture and Orchestration. IEEE Commun. Surv. Tutor. 19, 1657–1681.

Tan, L.T., Hu, R.Q., Member, S., 2018. Mobility-aware edge Caching and Computing in Vehicle networks: A deep reinforcement learning. IEEE Trans. Veh. Technol. 15.

Ur Rehman, C.S., Wah, T.Y., Iqbal, A., Jayaraman, P.P., 2016. Opportunistic computation offloading in mobile edge cloud computing environments. In: 17th IEEE International Conference on Mobile Data Management. IEEE, pp. 208–213.

Žliobait, I., 2010. Learning under Concept Drift: An Overview. pp. 1–36, arXiv:1010.4784.

Wang, Y., Chen, I.R., Wang, D.C., 2015. A survey of mobile cloud computing applications: Perspectives and challenges. Wirel. Pers. Commun. 80, 1607–1623.

Wang, Q., Guo, S., Liu, J., Yang, Y., 2019. Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing. Sustain. Comput.: Inform. Syst. 21, 154–164.

Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., Chen, M., 2018a. In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication By Federated Learning. https://arxiv.org/abs/1809.07857.

Wang, Y., Sheng, M., Wang, X., Wang, L., Li, J., 2016. Mobile-edge computing: Partial Computation Offloading using dynamic Voltage Scaling. IEEE Trans. Commun. 64, 4268–4282.

Wang, S., Zhang, X., Zhang, Y., Wang, L., Yang, J., Wang, W., 2017. A survey on mobile edge networks: Convergence of Computing, Caching and Communications. IEEE Access - Spec. Sect. Secur. Anal. Intell. Cyber Phys. Syst. 5, 6757–6779.

Wang, Z., Zhong, Z., Ni, M., 2018b. Application-aware offloading policy using SMDP in vehicular fog computing systems. In: 2018 IEEE International Conference on Communications Workshops, ICC Workshops 2018 - Proceedings. pp. 1–6.

Wang, Z., Zhong, Z., Ni, M., 2018c. Application-aware offloading policy using SMDP in vehicular fog computing systems. In: IEEE International Conference on Communications Workshops, (ICC Workshops'18). IEEE, pp. 1–6.
Wu, Q., Ge, H., Liu, H., Fan, Q., Li, Z., Wang, Z., 2020. A Task offloading scheme in vehicular Fog and cloud computing system. IEEE Access 8, 1173–1184.
Yazici, M.T., 2018. Edge machine learning: Enabling smart Internet of Things applications. Big Data Cogn. Comput. September, 17.
Yi, S., Li, C., Li, Q., 2015. A survey of Fog Computing: Concepts, Applications and Issues. In: Workshop on Mobile Big Data - Mobidata '15. pp. 37–42.
Yu, S., Wang, X., Langar, R., 2017. Computation Offloading for Mobile edge computing: A deep learning approach. In: IEEE 28th Annual International Symposyum Personal, Indoor, Mobile Radio Communication. pp. 1–6.
Zhang, K., Mao, Y., Leng, S., Maharjan, S., Zhang, Y., 2017. Optimal delay constrained Offloading for Vehicular edge computing networks. In: 2017 IEEE International Conference on Communications. pp. 1–6.
Zhang, Y., Niyato, D., Wang, P., 2015. Offloading in Mobile Cloudlet systems with intermittent connectivity. IEEE Trans. Mob. Comput. 14, 2516–2529.
Zhou, Z., Chen, X., Li, E., Zeng, L., Zhang, J., 2019. Edge intelligence: Paving the Last Mile of artificial Intelligence with edge computing. Proc. IEEE 107, 1738–1762.

Gonçalo Carvalho received the B.Sc. in Geography from University of Coimbra (UC), the M.Sc. degree in Geographical Information Systems from University of Trás-os-Montes e Alto Douro (UTAD) and B.Sc. in Informatics Engineering from Polytechnic of Coimbra (IPC), institutions located in Portugal, in 2005, 2009 and 2016, respectively. After the experience as a web and software developer between 2015 and 2018, he currently has a studentship for his Ph.D. research from the University of Coimbra, under the scope of "5G - Components and Services for 5G Networks" (POCI-01-0247-FEDER-024539). His main research interests are in the areas of data bases, distributed systems, edge computing and machine learning.

Bruno Cabral concluded his Ph.D. with honors in 2009 at the University of Coimbra (UC) in the area of Informatics Engineering. Bruno holds a Tenured Professor position with the Informatics Engineering Department of the University of Coimbra (UC). Bruno has been an Adjunct Associate Teaching Professor at the Carnegie Mellon University (CMU), USA, and a faculty of the Dual-degree Masters in Software Engineering (MSE). He is the coordinator of the Master Program in Software Engineering at the UC and a senior researcher of the Systems and Software Engineering group of Centre of Informatics and Systems of the University of Coimbra (CISUC). His main research interests are in the areas of distributed systems, parallel programming languages, machine learning, and dependable computing. Bruno was either the PI or staff member on multiple EU, Government and privately funded research projects, and frequently works as a consultant to the software industry.

Vasco Pereira received his Ph.D. in Informatics Engineering in 2016, from the Faculty of Sciences and Technology of the University of Coimbra (Portugal). He is currently an Assistant Professor at the Department of Informatics Engineering at the same university and the vice-coordinator of the bachelor's degree in Informatics Engineering. He is also a researcher at the Laboratory of Communications and Telematics of the Centre of Informatics and Systems of the University of Coimbra (CISUC) where he has been involved in national and European research projects. He is currently involved in projects such as 5G - Components and Services for 5G Networks (POCI-01-0247-FEDER-024539), MobiWise - from mobile sensing to mobility advising (P2020 SAICTPAC/0011/2015) and SOCIALITE - Social-Oriented Internet of Things Architecture, Solutions and Environment (POCI-01-0145-FEDER-016655). His main research interests include QoS, performance in Wireless Sensor Networks and IoT. His homepage can be reached at http://faculty.uc.pt/uc26416.

Jorge Bernardino is a Coordinator Professor at the Polytechnic of Coimbra - ISEC, Portugal. He received the Ph.D. degree from the University of Coimbra in 2002. His main research interests are Big Data, NoSQL, Data Warehousing, Software Engineering, and experimental databases evaluation. He has authored over 200 publications in refereed journals, book chapters, and conferences. He was President of ISEC from 2005 to 2010 and President of ISEC Scientific Council from 2017 to 2019. During 2014 he was Visiting Professor at CMU - Carnegie Mellon University, USA. He participated in several national and international projects and is an ACM and IEEE member. Currently, he is Director of Applied Research Institute of the Polytechnic of Coimbra, Portugal.