Machine Learning Meets Computation and Communication Control in Evolving Edge and Cloud: Challenges and Future Perspective

Tiago Koketsu Rodrigues, *Member, IEEE*, Katsuya Suto, *Member, IEEE*, Hiroki Nishiyama, *Senior Member, IEEE*, Jiajia Liu, *Senior Member, IEEE*, and Nei Kato, *Fellow, IEEE*

***Abstract*—Mobile Edge Computing (MEC) is considered an essential future service for the implementation of 5G networks and the Internet of Things, as it is the best method of delivering computation and communication resources to mobile devices. It is based on the connection of the users to servers located on the edge of the network, which is especially relevant for real-time applications that demand minimal latency. In order to guarantee a resource-efficient MEC (which, for example, could mean improved Quality of Service for users or lower costs for service providers), it is important to consider certain aspects of the service model, such as where to offload the tasks generated by the devices, how many resources to allocate to each user (specially in the wired or wireless device-server communication) and how to handle inter-server communication. However, in the MEC scenarios with many and varied users, servers and applications, these problems are characterized by parameters with exceedingly high levels of dimensionality, resulting in too much data to be processed and complicating the task of finding efficient configurations. This will be particularly troublesome when 5G networks and Internet of Things roll out, with their massive amounts of devices. To address this concern, the best solution is to utilize Machine Learning (ML) algorithms, which enable the computer to draw conclusions and make predictions based on existing data without human supervision, leading to quick near-optimal solutions even in problems with high dimensionality. Indeed, in scenarios with too much data and too many parameters, ML algorithms are often the only feasible alternative. In this paper, a comprehensive survey on the use of ML in MEC systems is provided, offering an insight into the current progress of this research area. Furthermore, helpful guidance is supplied by pointing out which MEC challenges can be solved by ML solutions, what are the current trending algorithms in frontier ML research and how they could be used in MEC. These pieces of information should prove fundamental in encouraging future research that combines ML and MEC.**

***Index Terms*—Mobile edge computing, cloudlet, scalability, communication/computation resource management, ML, artificial intelligence.**

I. Introduction

CLOUD Computing [1], [2] is a service model where user devices are able to connect to cloud servers. The servers have a significantly wide pool of resources (such as communication bandwidth, energy reserves/sources, processors, computing capability, computer memory), a lot more than the user devices. Thus, the client devices can use those resources to execute more demanding jobs and tasks whose requirements go beyond what can be done locally by sending these tasks to the servers. The cloud denomination comes from the offloading to remote machines and the transparency of the service. Forbes [3] predicts that the cloud computing market will reach US$411 billions by 2020, and many big companies already have established cloud services, such as Google [4], Amazon [5], and Microsoft [6].

Cloud computing can work with remote servers, located far away from the user device, which could be a static desktop computer in the case of Conventional Cloud Computing [1], [2] or a mobile device in the case of Mobile Cloud Computing [7], [8]. Mobile devices make the situation significantly different. Desktop computers sometimes can use cabled connections to reach the cloud servers, whereas mobile devices more often than not must depend on wireless and cellular networks in order to connect to the Internet and the cloud servers. Additionally, mobile devices are even more limited than desktop environments, with fewer resources and a smaller array of applications that can be executed locally [9]. This also means that they have more to benefit (i.e., the increase in the number of possible applications will be bigger) from cloud computing than desktop computers. However, the need to use wireless communication and even the device's limitations complicate the deployment and the implementation of cloud computing services. To make matters more difficult, many mobile devices and mobile applications, such as Virtual / Augmented Reality programs [10], Tactile Internet applications [11], and Internet of Things (IoT) [12]–[14],

Manuscript received January 17, 2019; revised June 7, 2019 and August 12, 2019; accepted September 21, 2019. Date of publication September 24, 2019; date of current version March 11, 2020. *(Corresponding author: Tiago Koketsu Rodrigues.)*

T. K. Rodrigues and N. Kato are with the Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan (e-mail: tiago.gama.rodrigues@it.is.tohoku.ac.jp; kato@it.is.tohoku.ac.jp).

K. Suto is with the Graduate School of Informatics and Engineering, University of Electro-Communications, Tokyo 182-0021, Japan (e-mail: k.suto.jp@ieee.org).

H. Nishiyama is with the Graduate School of Engineering, Tohoku University, Sendai 980-8579, Japan (e-mail: hiroki.nishiyama.1983@ieee.org).

J. Liu is with the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: liujiajia@xidian.edu.cn).

Digital Object Identifier 10.1109/COMST.2019.2943405

TABLE I
TABLE SUMMARIZING THE MAIN CHARACTERISTICS OF THE DIFFERENT CLOUD COMPUTING MODELS

Category	Server Location	Server Capacity	Number of Servers	Client Profile	Scenario Dynamicity
Conventional Cloud Computing	Distant (backbone)	High	Few	Fixed	Low
Mobile Cloud Computing	Distant (backbone)	High	Few	Mobile	Medium
Edge Cloud Computing	Near (edge)	Low	Many	Fixed	Medium
Mobile Edge Computing	Near (edge)	Low	Many	Mobile	High

need ultra-low latency, which cannot be possibly delivered by Mobile Cloud Computing due to the distance between the user devices and the cloud servers [15]–[18], which can sometimes be located in different continents [19]. To enable the execution of these applications, there have been recent efforts to create an Edge Cloud and Mobile Edge Computing (MEC) [20], [21],[1] where the cloud servers are, instead of deployed remotely, installed on the edge of the network, near the users.[2] In order to make this possible, there must be an edge cloud server near all users, since the main point of MEC is a close-range connection between the mobile device and the server. However, this means a huge density of cloud edge servers, which incurs an incapacitating financial cost. To counterbalance this, these edge cloud servers are designed to be less powerful than the remote cloud servers, which are more concentrated but fewer in number. Due to this smaller capacity, they have been denominated cloudlets [27], [28]. In summary, MEC offers lower latency and allows mobile devices to execute more demanding applications.

The possibility of using real-time applications and lifting device limitations has made MEC and cloudlets to be considered for future network designs along with the IoT [29]–[31] and 5G Networks [25]. However, there is another complicating issue in this caused by a key characteristic of IoT and 5G: a massive amount of devices. National Instruments estimates that 20 billions devices will be connected through 5G by 2020 [32], while Ericsson predicts 18 billion devices connected to IoT by 2022 [33]. Such scale cannot even be handled by Conventional Cloud Computing alone, stressing even further the need for MEC [14], [34]. But, even with low quantities of devices, many problems related to MEC are too complex to be solved [35]. With this massive amount of devices also comes a great variety of services and applications. Designing a MEC environment that takes into account the various kinds of wireless characteristics, device capabilities and service requirements is yet another layer of difficulty that cannot be solved by existing solutions and heuristical algorithms [36]–[38].

Table I summarizes the most important points between different types of cloud computing service. Note how the location of the server and the type of device vary between them. In this paper, we will focus on MEC and how to solve its issues of problem dimensionality, number of parameters and solution executability. Nonetheless, the considerations offered here could also be applied to the other service models due to the similarities between all of them.

One alternative for dealing with these issues of parameter dimensionality and algorithm feasibility in MEC is to use Machine Learning (ML) solutions [39], [40]. The base behind ML is to allow and enable the computer program to analyze and draw conclusions on the data by itself. The program is capable of doing this by learning the intricacies of the problem it is attempting to solve and making data-driven predictions and decisions, progressively improving its performance in one pre-determined task [41], [42]. In this area, the programmer's job is to allow the program to learn the problem (i.e., train the program), which can be done by feeding it historic input and output, or, in other words, what is the output (performance) X of solution Y given the input Z. Given a large enough amount of tuples (X, Y, Z) and a method for evaluating the value of X, the program should be capable of drawing patterns and eventually creating efficient solutions [43], [44]. This definition is very generic and broad and there are many variations of it, but the fundamental is the same: ML programs are capable of finding solutions that could not normally be devised (or sometimes even understood) by its programmers. These solutions, albeit not necessarily optimal, are usually efficient enough. Moreover, and more important in this discussion, current convex optimization[3] techniques applied to MEC rely on relaxation methods in order to deal with the high dimensionality of the realistic scenarios, i.e., they cannot be applied to the original problems [35], [41], which can only be tackled by ML [45]. Besides that, ML is also a very suitable option for dynamic scenarios, since ML models can find efficient solutions even with small changes in the problem, whereas conventional approaches may need a new execution.

The difficulties in designing solutions for MEC and the advantages of ML motivated us to write this paper. Here, we present a thorough survey of works utilizing ML in MEC and an instructive guide on how to utilize that paradigm in this environment. We hope to motivate future research in this area, index existing works, and provide support for the evolution

[1]Special note here to Fog Computing [22], [23], a term defined by the OpenFog Consortium [24] which denotes, similarly to MEC, a paradigm where cloud computation is moved closer to the origin of the data and the requests, at the edge of the network. However, Fog Computing is usually utilized specifically for deploying servers in Local Area Networks gateways.

[2]It is important to say here that MEC is also used to denote Multi-Access Edge Computing, a term coined by the European Telecommunications Standards Institute [25], [26] to denote the usage of edge cloud servers by mobile networks as well as Wi-Fi and other fixed access technologies. Multi-Access Edge Computing is a more broad term, encompassing Edge Cloud Computing and Mobile Edge Computing, but we opted to go with mobile over multi-access to emphasize the dynamicity of the scenarios seen in the literature. Thus, MEC in this manuscript refers to Mobile Edge Computing. Nonetheless, many of the considerations found here can be applied to Multi-Access Edge Computing as well.

[3]ML itself is also used for optimization. Thus, to avoid confusion, we will use "convex optimization" when talking about the mathematical, more conventional variety of optimization and simply "optimization" when talking about learning-based techniques for optimizing a function.

TABLE II
EXISTING SURVEYS THAT ARE RELATED TO THE PRESENT WORK, SHOWING THE APPLICATION OF ML FOCUSING ON SOME AREA OF COMPUTER NETWORKS

Survey	Focus point
P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza [46]	Self-organization in cellular networks
U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley [37]	Integration with IoT and sensor networks
S. Das and M. J. Nene [38]	Intruder detection and identification of malicious behaviour in networks
R. Mishra, P. Verma, and R. Kumar [47]	Gateway discovery in Mobile *Ad-Hoc* Networks
N. Z. binti Zubir, A. F. Ramli, and H. Basarudin [48]	Medium access in sensor networks
M. A. Alsheikh, S. Lin, D. Niyato, and H. P. Tan [49]	Applications in sensor networks and related solutions and challenges
M. Amiri, and L. Mohammad-Khanli [50]	Application demands in conventional cloud computing
M. Demirci [51]	Energy efficiency in conventional cloud computing
X. Fei, N. Shah, N. Verba, K.-M. Chao, V. Sanchez-Anguix, J. Lewandowski, A. James, and Z. Usman [52]	Cyber-Physical Systems

of future networks that utilize cloud computing. Given that objective, we can state the major contributions of this work as follows:

- The main problems of MEC, i.e., the challenges that should be addressed and solved in order to guarantee high quality and efficient operation, are also listed, explained, and categorized. This should aid not only researchers interested in ML solutions but also those working with MEC in general when identifying which problem to tackle.
- Additionally, we provide a network design and communication-focused explanation of ML algorithms that currently trending in research and industry in order to make researchers comfortable with the topic and to promote this paradigm as an important tool for future network orchestration. Obviously, this point is of use even to researchers working outside of MEC.
- We summarize recent and significant works in the literature that utilize ML together with MEC, showing which algorithms they chose to utilize with their selected problems.
- Finally, new future directions for MEC research that are enabled by ML are shown as encouragement for future research works.

Existing surveys on MEC [25], [53]–[57] traditionally focus on establishing MEC, how to implement it, and offer heuristic solutions for network design and usage with 5G and IoT. Moreover, as shown in Table II, existing network-focused surveys on ML [37], [38], [46]–[52] talk about applications in generic networks or discuss how ML can deal with problems that have too much associated data in certain networks, but not MEC. Unlike the aforementioned literature, this paper specifically focuses on how to use ML to deal with problems characteristic to MEC, presents what techniques work better for each MEC challenge, and offers a guide on how to apply ML to future research on MEC.

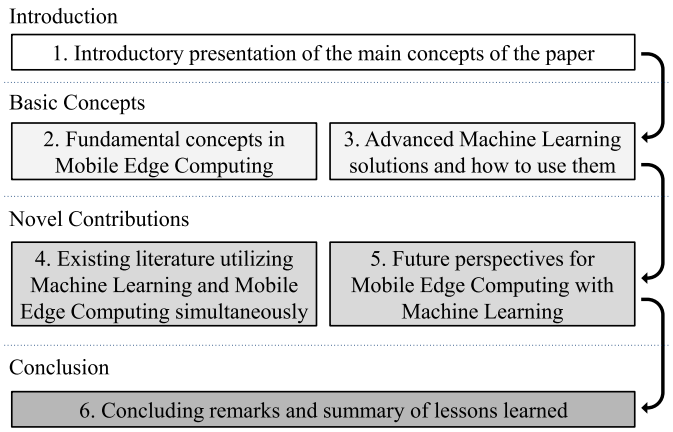

Fig. 1. How the sections in this paper relate to each other.

The remainder of this paper is organized as follows. Section II presents ideas on how to model MEC and lists related research problems. Section III introduces specific algorithms of the ML paradigm, detailing their mechanisms, exposing their strengths as well as their weaknesses, and explaining how they can be adapted for MEC. Section IV shows a survey of existing research work that utilizes ML when addressing MEC problems, detailing which problem they tackle and what solution was chosen to do it. Section V offers future directions for MEC research and how ML solutions could be integrated into each one of them. Finally, Section VI concludes the paper by summarizing the lessons learned. The initial sections introduce the main areas of this paper in order to familiarize less experienced readers while the last sections join all information into a survey and concluding remarks. Fig. 1 illustrates the interdependency and connection between the sections in this paper; sections in each layer of the figure depend on all sections in the layers above, as represented by the arrows.

II. MOBILE EDGE COMPUTING MODELING AND CHALLENGES

In this section, we will present the usual MEC service model that is generally utilized in the literature. This service model can be configured, where configuration is the set of options and parameters that operators can decide in order to deliver the best service possible following some pre-determined evaluation. To find these optimal parameters, some challenges and problems related to MEC have to be solved; such issues are also explored in this section, but not before a presentation of existing surveys on MEC.

A. Service Model

As a prologue to the service model, it is important to explain the main entities in MEC [20], [27], [28], [58]. User Device (or user) is the one who contracts the service and will create the tasks (or input, or requests) that must be executed by the MEC system; additionally, the results (or output) of the

tasks must be sent back to the user device after it is done. Access Point is the device that is used by the User Device to connect to the network and reach the MEC system, such as a base station with a cellular antenna. The User Device must utilize the Access Point to communicate with the other entities and the connection between Access Point and User Device is usually wireless. Cloudlet (or edge cloud server) is a physical computer that works as a server located at the edge of the network. The Cloudlet contains the resources needed for executing the requests created by the User Device. Compared to Conventional Cloud Servers, Cloudlets have lower capabilities (in computation, communication, etc.), hence their name. Virtual Machine (or virtual server) is a virtual environment located inside the Cloudlet, result of virtualization of the resources of the Cloudlet. This allows for the creation of multiple "servers" even if there is only one single physical computer, making the separation of resources and their management easier. The Virtual Machine receives the requests by the User Device, executes the requests, and sends back the output. In order to execute these tasks, the physical server (e.g., Cloudlet) allocates some of its resources to the Virtual Machine according to the demands of the requests and a pre-determined policy. Each User Device is associated with a single Access Point, a single Cloudlet and a single Virtual Machine. Generally speaking, each Virtual Machine is only associated with one User Device (thus, users cannot send their requests to other virtual machines, and the requests must be routed to the host of the virtual machine, regardless of where the user is and whether the user moved to somewhere since the creation of the virtual machine), while Access Points and Cloudlets can be associated with many User Devices. In physical terms, Cloudlets are usually located in the same location as Access Points, for convenience [27], [59]–[61].

Also of note is that in MEC (and cloud computing as a whole), there are different products that can be offered as a service. Those are divided between Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [62]. In SaaS, the whole application is offered by the cloud service provider, and the users just supply the data that serves as input and in return receive the corresponding output [63]. In PaaS, the service provider offers everything up to the operating system and the client provides the application that will be executed [64]. Thus, in PaaS, the user will send both its application and the input for it, the application will utilize the cloud resources, and then the user will collect the output. Finally, in IaaS the service provider makes available to the user its computers, storage, networking equipment through virtualized resources that the client can acquire as needed, while the user is responsible for deploying everything from the operating system onward [65]. In other words, the user provides and manages the platform, that will be run on top of the service provider's resources. Note, however, that all these models operate with virtual servers on top of the physical resources from the provider, as shown in Fig. 2. Thus, the structure based around the virtual server described previously can still be applied to MEC regardless of the product being offered.

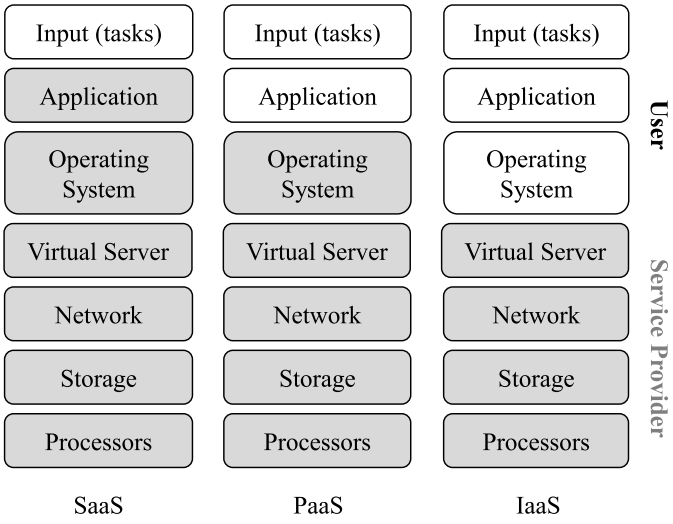

Fig. 2. Comparison between SaaS, PaaS, and IaaS, specially in what is offered by the users and what is offered by the Service Providers in each service model.

The MEC service model is usually assumed to be as follows [21], [27], [66]–[70]. When first entering the system, the user device makes an initial request that is received by the closest cloudlet. This request may be forwarded to a remote centralized entity (a central device with full information and control of the MEC system) or resolved in that edge cloud server. The result of this request is the creation of a Virtual Machine which will serve that user[4] (realistically speaking, there is an overhead related to virtualizing the resources of the host server when creating the virtual machine, which delays the beginning of the service by an amount that varies depending on what exact service is offered [71]). As for what this service entails, it depends on the application, but it could go from task offloading and extending the environment of the user to even content delivery and caching, as mere examples. This virtual server will be hosted by one of the cloudlets of the system, with the choice of which one being made following a pre-determined policy. After this initial setup, the user will proceed to send tasks to its corresponding virtual server, who will receive this input and process them utilizing the local resources of the edge server. After finishing execution, the virtual server sends the output of the task back to the device, which displays it to the user as if it was a terminal. This communication between user device and edge cloud / virtual server happens wirelessly between the device and the access point and through a cable between the access point and the edge server. Fig. 3 shows a diagram illustrating a typical MEC service model and all of its entities (note how the virtual server is inside the cloudlet). There are some relevant variations to this model, with the most prevalent being: multiple virtual servers [72], [73], task fragmentation

[4]The Virtual Machine is intrinsically connected to the service provided. Thus, if MEC is offering a specific application as a service, the Virtual Machine is capable of performing any routines related to that application. Conversely, if the platform or infrastructure is offered as a service, the virtual machine can execute code sent by the user.

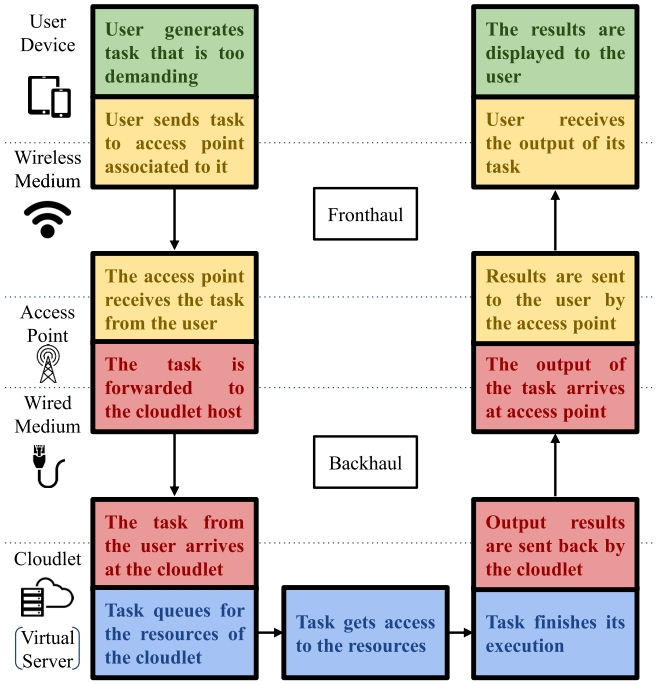

Fig. 3. Diagram illustrating a generic view of the service model in MEC (although Access Point and Cloudlet are virtually different entities, they are usually physically together and always at the edge).

[74], [75], virtual server migration [66], [76], cooperation with remote servers [77], [78], and virtual containers [79], [80].

As a minor sidenote, one service model variation that deserves more attention is user mobility. The decision to consider user movement in the assumed scenario can indeed increase significantly the difficulty to solve a problem [81]. The issue comes from the uncertainty and dynamicity created by user mobility since the channel proprieties are always at risk of changing due to the user moving [61], [73], [82]. This movement could simply put more distance between the user and the access point, decreasing signal quality due to path loss, or even put the user in an area with more interference and obstacles. Consequently, it is very much a possibility that the user will transition to a state where a new access point is more favorable, which further complicates things with handover between the base stations. Moreover, it may also mean that the user suddenly benefits more from connecting to a new edge server. Finally, all these changes are very difficult to predict, even if the users are assumed to follow some mobility models. Thus, a lot of MEC research prefers to ignore user mobility altogether [74], [77], [83], which is an option albeit not the best one in terms of choosing a realistic scenario.

As it is evident from the model and regardless of variations, service in MEC can be divided into two main segments: the transmission element (the communication between the user and the cloudlet) and processing element (the execution of the request sent by the user to the cloudlet) [21], [66], [67]. In the communication side, performance is mainly affected by interference between devices, bandwidth available, physical propagation, noise, transmission power and payload size [13], [74], [82], [84]. Meanwhile, in the computation side, performance mainly depends on server processing speed, size of the task (number of instructions, cycles required, etc.) and the competition for server resources [78], [85]–[87]. Some MEC applications will generate heavier communication payload (e.g., content delivery, where users may download videos and pictures that will put a lot of stress in the network), some MEC applications need to execute many instructions for their requests and will demand a lot of time from the processors at the cloud servers (e.g., image processing, where the images sent by the user devices must be fully analyzed while looking for points of interest), while some other MEC applications are a mix of both (e.g., big data analytics, where big databases must be transmitted and then analyzed at the processors). These show examples of high communication burden and high computation burden among MEC applications. These different profiles and requirements further complicate MEC operation, as the applications will need different resources and systems if an efficient service is desired (e.g., for content delivery, investment in bandwidth is desired, while for image processing, investment in more capable servers is desired; for servicing both simultaneously, either the network is heavily and expensively equipped or the algorithm must be very efficient in resource allocation, which is obviously preferred) [88].

Additionally, it is worth noting that the service model in MEC is virtually identical to the one in Mobile Cloud Computing and Conventional Cloud Computing, involving Virtual Machines and job offloading. Consequently, many of the design and implementation problems in Mobile Cloud Computing and Conventional Cloud Computing are similar to what we find in MEC. However, there are some key differences. Firstly, due to lower latencies, MEC can service real-time applications and devices that demand quick response times on top of all applications and devices that conventional cloud computing handles, meaning that MEC has to work with a higher variety and bigger amounts of clients. Secondly, not only there are more clients, mobile devices themselves have features that cannot be ignored which may make local processing impossible or at least undesirable, such as their battery level, memory, and transmission data rate [31]. Those resources are particularly limited in mobile devices and their availability may change over time, which makes decisions on offloading, for example, even more complicated. Also, MEC also has to work with many more servers (actually, the concentrated servers in Mobile Cloud Computing can even be considered as one single giant server [89]). Finally, the networks in the edge are much more dynamic. These characteristics mean that problems in MEC have more parameters, higher dimensionality and incur higher volumes of data. For these reasons, solutions used in Mobile Cloud Computing / Conventional Cloud Computing often do not apply to MEC.

B. Challenges

With an objective function (or a combination of more than one [74], [90]) chosen, the cloudlet or the central office of the MEC service must then decide which configuration to utilize in order to optimize this objective. This configuration must answer possible questions regarding the service model of the

system (e.g., which cloudlet will host the virtual server, which jobs will be offloaded). In the literature, this translates to a set of problems that are solved in order to raise the efficiency of the MEC system. These problems are usually independent of service variation or objective function. In the following subsections, we present categories of notable MEC problems.

1) Offloading Decision: Given one of the objectives previously mentioned, the class of problems related to offloading decision decides where the user-generated tasks should be executed. The choices can be local, inside the user device itself; the edge cloud servers, located near the user, following MEC; and the remote conventional cloud servers (as in Mobile Cloud Computing). In case of choosing the edge cloud servers or the remote cloud servers, a second possible choice is which server will execute the task, although this can be omitted in both cases (e.g., no virtual server migration so the host server is already decided; the remote cloud servers operate jointly as a massive super server [85], [89]). Moreover, the whole definition of the user task can vary between works, whereas some consider solid, indivisible tasks [85], [91] while others fragment them into subtasks (that can be dependent or independent between themselves) [74] and some others even duplicate tasks for redundancy [92]. Despite all these possible variations, the pattern is the same: decide where the user-generated task will be executed such that the objective function is optimized. Research work in this area usually concludes that offloading decision (even in simplified scenarios with a single user or a single server) is either NP-hard [74] or NP-complete [85], such that relaxation or heuristic algorithms are usually the proposed solutions. However, such algorithms have their complexity order proportional to both the number of devices and servers, which renders them unfeasible in the future networks with massive amounts of agents.

2) Resource Allocation: Even if the offloading target is decided, service may be affected and changed by allocating more or fewer resources to the user. Because in real-life situations, both the user and the cloud server, either at the edge or at a remote location, have a limited amount of resources, the decision of how many to allocate to the MEC service is an important one in order to operate the network efficiently and to unleash the full potential of the system [77]. This resource allocation can be done either in the fronthaul, i.e., directly involving the user and its relation with the rest of the system, or in the backhaul, i.e., without direct relation to the user. The resource allocation decision can be made by a centralized office, with full [77], [86] or partial knowledge [93] of the system, or individually by each server / user device [13], [94]. These resources are mostly related to the communication side (sending input and output between the user device and the server), although there are some mentions to the applicability in the computation side (the execution of the task). For communication, examples are channel bandwidth [77], [86] and transmission power [66], [67], among others. All these affect the communication latency between the user and the server, so they consequently have effects on the service latency. Transmission power has a direct effect on the energy consumption of the nodes, while the communication delay itself has an indirect consequence in that the nodes may have to spend more time transmitting. The same can be said about profit, relating the money spent on energy and the monetary gain by finishing more tasks in less time. Regarding resources in the computation side, notable examples are number of processors in a single server [77], [86] and even the speed of these processors [74], although research involving such parameters is rather scarce. Specifically related to backhaul resource allocation, notable mentions are servers sharing workload [84], servers sharing resources [95], virtual servers being migrated [73], [82], among others. The problem of deciding how many resources to allocate in order to optimize a system-wide metric is also usually solved by heuristic algorithms that are usually more complex if the model of the scenario is more realistic. Furthermore, since this allocation is taken with a system view, it is also affected by the number of devices and servers, which renders them unfeasible in the future networks with massive amounts of agents.

3) Server Deployment: A problem that is very characteristic of MEC is the choice of where the servers will be deployed. Being edge servers, there are usually multiple choices of where they could be installed. As mentioned before, they are typically located near network access points, but since it is not viable or necessary to have one server in each access point, there is still a choice to be made [59], [60]. The conclusion is that there must be a choice of where to deploy the servers, which naturally should be guided by an objective function, trying to optimize a pre-determined goal, such as the ones listed in our previous sub-section. This usually means deploying servers near places where users conglomerate, so more clients can be served (raising profit) with shorter connection distance [60], [96]. However, in MEC particularly, users can move, which may turn a deployment location from a desirable one into a bad position [61]. Moreover, differently from virtual machines, server deployments are usually permanent, or at the very least expensive to change, demanding extra care when choosing. Moreover, the deployment choice also comes with important financial consideration, as the number of servers and their locations all come with associated costs [97], [98]. The deployment places may have to be rented or equipment has to be acquired, making this an essential consideration even if the objective function is not based on the service provider's profit. From this discussion, it is clear that solutions to the deployment challenge depend heavily on how many servers will be deployed and how many choices are there, which in itself already means a complex scenario. Furthermore, the dynamicity of MEC means that the scenarios keep changing, which comes in contrast to the permanency of the deployment decision, making the choice of a good location even more difficult, especially in a future with the massive amount of servers and devices to be served.

4) Overhead Management: Finally, another point of note is how all these configurations and actions incur significant overhead. The choices of where to offload and how many resources to allocate are all performed online, meaning that until such decisions are made, the received requests will not be processed nor answered, increasing the latency and decreasing service quality [99], [100]. Moreover, there are other actions

in the service model of MEC that result in overhead as well, particularly the ones related to the virtual server. Not only a choice has to be made of which physical server will host the virtual server for each client, the virtualization process itself and the initial setup of the virtual server takes time [71]. Additionally, in scenarios with virtual server migration, this transfer of the virtual machine between servers also takes time, during which the service is potentially paused [72]. Other possible causes for overhead cost include handover between networks when the user moves [101], [102], orchestrating the cooperation between different domains [19], [84], deciding the routing between the user device and the associated servers [103], [104], among others. Most works tend to ignore some if not all of these overhead sources, but for a more realistic modeling of the scenario, they should be considered. However, not only the calculation of overhead itself is quite difficult, overhead costs themselves will probably ramp up as more devices and more servers are considered since these extra agents mean more complications for the algorithms responsible for calculating the configurations, which means they will take longer to execute [36], [38].

Lessons Learned: This section offers the tools for modeling problems in MEC. This includes the basic service model used in literature, with its entities and variations. More importantly, this section presents broad categories of MEC challenges, which are a useful starting point for researchers, who could use this to identify the theme and goal of their next work. In this sense, it is important to also discuss what are the current directions and trends in each challenge. For example, in the Offloading Decision category, the final goal is obviously to utilize multiple and various servers, in the edge or not, as possible destinations [9], [27]. This means a lot of variables to consider, as discussed previously. Compounded by a high quantity of users, the resulting problem ends up being very difficult to solve. Thus, more and more literatures pieces published recently are using a user-centric approach, where they focus on solving the problem for a single user [94], [105], [106]. This means a simpler (and thus feasible) solution that could be applied in a user-by-user case. This idea is also applicable to and can be seen in resource allocation, both in the fronthaul and the backhaul [53], [107]. However, for resource allocation, it is also notable how more and more research is being made combining both the Communication and Computation planes of the service model [93], [108]. Previously, a lot of MEC work would focus on a single aspect, e.g., how to minimize the transmission delay, how to maximize processor utilization [13]. However, both transmission and processing are important for the quality of service and they are also dependent on each other, so much so that modifying communication parameters will affect the computation part of MEC. Thus, it is necessary when building a realistic solution to consider the whole service model of MEC, despite how complex this is [88]. In the case of Server Deployment, it is interesting to investigate other deployment locations besides next to base stations. Obviously, this comes with the drawback of the connection between the edge server and the base station, to integrate it with the network, which must be considered in the final model and objective function. Nonetheless, there are some benefits to this, as there may be locations with lower monetary costs (e.g., cheaper rent) or better energy infrastructure to set up the edge servers, bringing benefits to the service provider [109]. Moreover, there can be already existing servers that perform different, local functions in the edge (e.g., servers in a company or university) that could secondarily be utilized as MEC servers [27]. These possibilities are not very touched in the literature. Finally, Overhead Management as a whole is not studied enough, so there is plenty of work that could be performed in this field yet [99], [100]. The estimation and calculation of the overhead cost associated with each action and the integration of this with the other decision (e.g., the offloading destination or resource allocation ones), including its impact in the objective function, are useful future perspectives. This goes especially in contrast with the many research works that mainly assume that their proposed solutions will either be performed offline always or will have negligible cost, which is not realistic in most scenarios [99], [100], [102].

III. Machine Learning Techniques

The previous MEC discussion showed that most research that tries to address them will have issues with complexity and solutions that would not scale well. This is particularly worrisome when you take into account the 5G and IoT paradigms [14], [34]. To address this issue, we believe that one of the best options would be utilizing ML algorithms [110]. This section will first justify the application of ML solutions in general, then it will show existing surveys on ML and finally present some important ML algorithms that have gained traction recently in research and how to apply them to MEC.

A. Justification

ML cannot guarantee optimality but it offers solutions with low complexity (i.e., feasible even with bigger scales of variables) that still reach efficient configurations [111]. This is mainly due to their "learning" mechanisms, where the algorithm learns from past experiences, either from configurations it tried itself and evaluated their performance (unsupervised learning), through training sets with known optimal solutions (supervised learning), a mix of those two (semi-supervised learning) or by rewarding solutions that are closer to a goal/punishing solutions that are farther (reinforcement learning) [41], [44], [112], [113]. Moreover, most ML algorithms are capable of balancing between exploration (i.e., trying configurations outside the known patterns in an attempt to find something better) and exploitation (i.e., following the already mapped out patterns) when looking for solutions [114], [115], which only increases their efficiency. Thus, we can summarize the main benefits of ML as capacity of dealing with high dimensionality problems, aptitude at identifying patterns, dynamic solutions that handle different scenarios efficiently [44], [111]. These benefits mean less operational expenditure and capital expenditure for the providers of MEC since less overhead is spent on calculating the best configurations for the system. This lower cost comes without a significant loss in configuration efficiency. Finally,

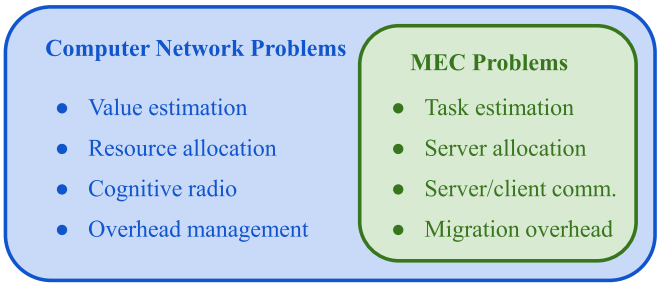

Fig. 4. Examples of problems that can be solved by ML in computer networks and MEC.

these configurations could be calculated by a centralized controller [116], [117] or in a distributed form [118], [119], with full [120], [121] or partial [122], [123] knowledge of the system, all depending on which ML algorithm is selected, i.e., there is a good ML solution for most situations [111], [124].

Table II shows many surveys on ML. Within those surveys, there are plenty of works that utilize ML to solve computer network issues, with examples of how ML can be used in management, configuration, and orchestration for general networks. For example, ML can be used to estimate important metrics by training ML models to produce the estimations based on the information available (e.g., historical logs created by the server about received tasks, connected users, etc. [125], [126]). One example is Channel State Information [127], which is a very important metric for evaluating connections and making decisions (such as to which base station to connect to and resource allocation) [128]. Another example is application demand, which is important for allocating the right amount of resources to satisfy such requirements [129], [130]. Beyond just estimating metrics, ML is also used for allocating resources directly, training the ML models to learn based on user features (e.g., user task requirement or connection information) how many resources should be given to each ideally [130]–[132]. Resources here could be server processors or communication bandwidth, for example. ML has also been widely applied to cognitive radio, for deciding transmission parameters (e.g., bandwidth and frequency) as the radio adapts and changes according to the present channel [133]. These examples just show how ML has been successfully applied to computer networks in general, meaning there is no reason to believe that MEC would be any different. In fact, many of the situations mentioned (i.e., metric estimation, resource allocation) apply to MEC. Edge cloud is just a bit more specific, with specific characteristics regarding edge servers and task offloading, as it was explained in the previous section in detail. Fig. 4 illustrates this point. Note how not only MEC problems are a subset of computer network problems but also MEC problems are more specific versions of the generic computer network problems.

As a final note, it is important to say that even ML solutions have problems with the number of users in MEC, given its massive numbers [44]. Such a high number of clients, and the resulting variables and parameters, make the learning process take longer. Nonetheless, it is also important to note that conventional heuristic algorithms and convex solutions have the exact same problems. In fact, such MEC solutions are often not even feasible [38], [41], [45], [108], [162]. Conventional solutions usually resort to relaxing the original scenario, resulting in a simpler problem but also one that is less realistic. By comparison, ML solutions can be applied to the whole scenario, with all of its variables, and still produce very efficient solutions with relatively quick execution times compared to conventional solutions.[5] For empirical evidence, there are plenty of examples in the literature of ML solutions outperforming conventional, non-learning solutions. A short list of this can be seen in Table III (for this table, please note that the improvement percentage is approximated unless the original paper gave exact values and best case scenarios are shown). Thus, despite it indeed being difficult to deal with the complexity and dimensionality of MEC problems, ML solutions are still better than conventional ones.

B. Advanced Techniques

In this subsection, specific ML algorithms will be recommended. Those were selected based on their usage in recent research works as well as their relevance in the industry. Efforts will be made to present their most relevant characteristics and particularly how to adapt them to MEC.

1) Deep Q-Learning Neural Networks: Neural networks [164], [165] are a classic ML algorithm where networks of perceptrons are built and organized into sequential layers. Each layer's input is connected to the output of the previous layer. In this sense, we have an input layer, directly connected to the features of our data; an output layer, for generating the result we want; and several hidden layers in between. The output of the neural network is evaluated by comparing with what the correct answer would be in case of supervised learning or with a general maximum/minimum objective function, depending on the problem. What is important is to have a loss function to compare the output of the neural network with the optimum result. This loss function is then utilized for looking for the best weight multipliers for each input for each perceptron in the network using Stochastic Gradient Descent [166], [167], in a process called backpropagation [168].

Nothing of this is new, as neural networks are well established and respected algorithms. Recently, however, with advances in processors and computers, it has been possible to build and operate neural networks with increasing amounts of hidden layers. This is significant because, although the number of layers is proportional to the complexity of the training of the network, more layers also allows the algorithm to create more and more relations between the features, allowing the neural network to solve more complicated problems [68]. These

[5]As some examples, Rodrigues *et al.* [21] utilized a ML algorithm that achieved less than 2% worse than optimal results while needing 2.31% of the time taken to calculate this optimal value mathematically (it is even mentioned that this is done in a simple scenario because, although the ML method could handle more complex cases, the mathematical method would take exponentially more time). Moreover, Yu *et al.* [163] utilized a different ML algorithm that achieved 97.23% accuracy taking 0.06% of the time taken by the convex optimization method. Obviously, these are just examples, but they should serve to illustrate how ML can achieve very close to optimal results at a tiny fraction of the time needed.

TABLE III
RESEARCH WORKS THAT USE ML SOLUTIONS AND THE DIFFERENCE THAT THEIR EVALUATION SHOWED WHEN COMPARED WITH OTHER, MORE CONVENTIONAL SOLUTIONS

Research Work	ML Algorithm	Comparison Algorithm	Metric	Difference
X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic [134]	Fuzzy control model [135]	Least recently used caching	Service delay	85% better
X. He, J. Liu, R. Jin, and H. Dai *et al*. [106]	Constrained Markov decision process [136]	Greedy/random mix solution	Performance and privacy tradeoff	50% better
A. Aral and I. Brandic [137]	Tree-based naive Bayes [138]	Logistic regression [139]	Availability prediction	1% better
T. Hou, G. Feng, S. Qin, and W. Jiang [140]	K-means clustering [141]	Least recently used caching	Cache hit rate and transmission cost	69% and 22% better
B. Li, K. Wang, D. Xue, and Y. Pei [96]	K-means clustering [141]	Density based clustering	Service delay	50% better
H. Du, S. Zeng, T. Dou, W. Fang, Y. Wang, and C. Zhang [142]	Clustering by Fast Search and Find of Density Peaks [143]	Density based clustering	Accuracy	65% better
T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato [21]	Particle swarm optimization [144]	Conventional greedy solution	Service delay	40% better
T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma [67]	Particle swarm optimization [144]	Conventional greedy solution	User capacity	180% better
J. Zhang, W. Xia, Z. Cheng, Q. Zou, B. Huang, F. Shen, F. Yan, and L. Shen [90]	Dynamic evolutionary game [145]	Conventional greedy solution	Energy consumption and service delay	35% and 20% better
H. Wang, R. Li, L. Fan, and H. Zhang [146]	Genetic algorithm [147] and simulated annealing [148]	Random allocation	Service delay	30% better
S. Kim, X. de Foy, and A. Reznik [76]	Simulated annealing [148]	Conventional greedy solution	Server usage and user capacity	15% and 18% better
J. Xu, L. Chen, and S. Ren [149]	Q-learning [150] and post-decision state learning [149]	Myopic optimization (no historic data)	Service delay	40% better
T. Yang, Y. Hu, M. C. Gursoy, A. Schmeink, and R. Mathar [151]	Deep Q-learning neural network [152]	Conventional greedy solution	Throughput	10% better
T. Q. Dinh, Q. D. La, T. Q. S. Quek, and H. Shin [153]	Model-free Q-learning [154]	Random offloading	Energy consumption	75% better
L. T. Tan and R. Q. Hu [155]	Deep Q-networks [156]	Random resource allocation	Throughput and cost	350% and 50% better
Q. Gao, L. Gao, T. Xue, X. Zhu, X. Zhao, and R. Cao [157]	Ant colony optimization [158] and multi-feature linear discriminant analysis [159]	RSCC (conventional solution)	Service delay	35% better
Y. Li and S. Wang [97]	Particle swarm optimization [144]	Conventional greedy algorithm	Server usage and energy consumption	15.4% and 11.5% better
S. Wu, W. Xia, W. Cui, Q. Chao, Z. Lan, F. Yan, and L. Shen [160]	Support vector machine [161]	Random offloading	Service delay	50% better

neural networks with many hidden layers are called "deep neural networks" [169] and they have attracted a lot of attention recently, from academia, industry and the media [170]–[172].

To compound with the potential of deep neural networks, there is a technique called Q-Learning [150]. Let's assume a state transition process, with a set of possible actions for each state with respective resulting state and reward for each action. Thus, there is a policy Q^* of actions for each state that maximizes the reward of the system. The Q-function for evaluating each policy Q utilizes the Bellman equation [173]. We can utilize this function and a deep neural network for finding the best action for each state, reinforcing choices that give more reward [156], [174]. Additionally, it is also advisable to balance between exploration (trying random actions) and exploitation (trying the action recommended by the neural network) when doing online training, which is the usual use case of deep Q-learning neural networks [152]. Interested readers are suggested to look into the references in this section for the details of how these networks and Q-learning work.

For MEC problems, it is important to model the scenario as a state transition process and describe each state with only the essential features, since more parameters will result in slower training. Personal experience can be used in this, as well as some algorithms, such as principal component analysis [175], [176] or linear discriminant analysis [159], [177]. For example, let's say we are looking for the best offloading destination for each task generated by our user, with a goal of minimizing the service delay. In this case, we can describe the state by the loads of all servers; the features of the generated task, such as the corresponding payload size and the amount of processing time needed; the delay to each server; and the position of the user if he is mobile. Then, from personal knowledge, we know that the load of the remote server is not relevant due to its higher capabilities. Furthermore, a feature selection algorithm like linear discriminant analysis tells us that user mobility is not very significant for service delay in our scenario, only altering the final result slightly. Thus, we remove these two features, making the problem more simple and speeding up the training of the neural network. Then, we iteratively select destinations for each task the user generates, alternating between using the neural network and selecting at random, and record the resulting service delay. This performance is then fed back to the neural network, which will reinforce choices that result in a lower delay. As an important side note, it is usually better to randomly explore the options in the beginning and slowly exploit more and more

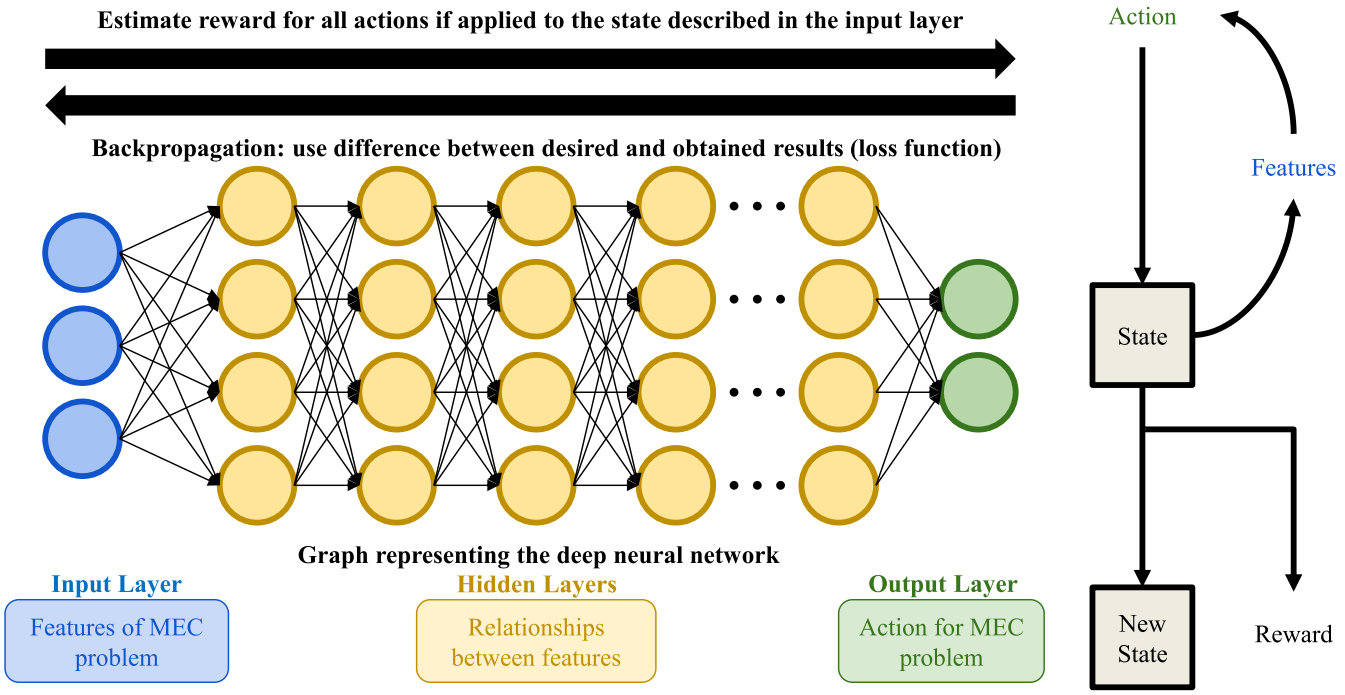

Fig. 5. Detailed specification of the most important elements in deep Q-learning neural networks.

through the neural network as it becomes more knowledgeable (which itself can be determined by performance or by the number of iterations). This will result into a poor service in the beginning, but as the neural network learns more and more, it will provide a lower service delay, balance the workload between servers and efficiently manage the system. Thus, the following guide can be used for modeling for deep Q-learning neural networks:

1) Determine what is the goal of the problem (e.g., achieving a high profit for the service provider).
2) Identify what is a state for your Mobile Edge Computing problem (e.g., bandwidth and processor allocation to each user).
3) Identify what changes the state of your scenarios (e.g., a user makes a new task request).
4) Identify possible actions (e.g., reject task request, allocate resources in one of the servers).
5) Determine the features related to the problem (e.g., resource allocation and task requirement).
6) Use feature selection techniques to remove unnecessary features, optimizing the algorithm.

For a more technical discussion, note that the algorithm must choose, with each state, whether to use the best policy found so far or search for a new one. Which choice is taken is randomly selected, but biased by a learning rate that determines the ratio between exploration and exploitation. If the algorithm goes with the best policy found so far, then the complexity involved is simply identifying the current state through its features and finding the corresponding best action indexed by the Q-learning module. At worst case, the complexity would be proportional to the number of features and the number of possible states. If exploration is chosen, then the algorithm must utilize the deep neural network module. At worst case, this is a complexity proportional to the total number of perceptrons in the network and the number of inputs for those perceptrons. As a caveat, it should be said the complexity itself is very implementation and problem-dependent, so these are just rough estimates [178]. It is also important to determine some hyperparameters. Notable ones are the learning rate, the number of hidden layers, the activation function of the perceptrons, the loss function, etc. These are problem-dependent and need to be tested and experimented with, although some good options are frequently used and can serve as a starting point [169], [179].

Fig. 5 illustrates the main points of deep Q-learning neural networks. The neural network component, symbolized by the graph of perceptrons in the figure, is utilized for estimating the reward of all possible actions, given the state described by the input layer. As thus, the loss function, used in backpropagation to improve the neural network, is based upon the difference between the estimated reward and the actual reward received by the action in that state. The loss function could be represented by cross entropy [180], for example. Thus, given the state we extracted the features from, we apply to it the best action as predicted by the neural network (or choose a random action, if exploration is desired). This will result into a new state and a resulting reward. We utilize this result to update the Q function of our problem. Additionally, the Q function is usually accompanied by a learning rate parameter to determine the weight of new results and old results and also a weight applied to the expected reward of future actions. Finally, by using the neural network to find actions and the Q function to evaluate the value of actions, we search for an optimal policy Q^* that will determine the best actions for all possible states.

The main advantage of deep Q-learning neural networks is the combined power of multiple hidden layers, which allows

for the neural network to identify and model complex relations between features, and the aptitude of Q-learning, an extremely efficient model for finding the optimal policy of actions [152], [179]. Additionally, while Q-learning by itself is not very practical nor realistic because it is limited to discrete variables and solutions, the neural network component of the algorithm allows for problems with continuous states and actions to be tackled. This allows for more complex functions and problems to be solved, such as the ones related to MEC [17], [151]. For our previous example, the function associated with an offloading decision problem of that nature is of such a high dimensionality that many hidden layers are needed to properly model it; a regular neural network would not be able to properly learn the problem because of this, resulting in inefficient solutions [169], [181]. Finally, Q-learning will be useful for facilitating the search for the optimal solution as well as introducing an efficient reinforcement learning method for adapting to real-life dynamic scenarios. In summary, this new flavor of neural network allows us to finally use this ML technique not only to properly solve MEC challenges but also to finally solve some of these more complex problems.

2) Stochastic Gradient Descent-Assisted Support Vector Machines: Support vector machines [161], [182] are another classic ML algorithm. As a quick summary for context, support vector machines, given a set of samples that belong or not to a pre-determined classification and that are plotted in a feature space, find a linear separator such that positive samples are on one side of the separator while negative ones are on the other side. Full mathematical details can be found elsewhere, but it is possible to find such a separator through Lagrangian multipliers in a way that the multiplier depends solely on the input samples and also such that the smallest distance between the separator and any sample is maximized. In possession of this separator, we can classify other inputs, even if they were not in our training set. Of note is also that even if the samples cannot be classified into positive/negative samples, the support vector machine can be used to optimize an objective function instead (but this comes with the drawback of not being able to use to the Lagrangian multipliers method). Additionally, even if such separator does not exist, the support vector machine can try to at least minimize the number of samples on the wrong side.

The main drawback with support vector machines is that the separator is linear, which results in poor performance if the underlying problem is not linear, something very common in MEC problems. The solution is to apply what are called kernels for mapping the feature space, and consequently the samples, into new dimensions [183]. Ideally, the samples can be linearly separated in this new space, such that the support vector machine can find a better solution in this kernelized dimensions. This is because a linear separator in the kernelized space is non-linear in the original space, allowing our algorithm to solve non-linear problems.

None of this is particularly new, and it actually does not perform really well because kernelized support vector machines ramp up in complexity the more training samples there are [183], [184], a huge drawback since this limits the precision of the algorithm. A relatively new solution to this is utilizing stochastic gradient descent on the samples in the kernelized space to accelerate the convergence of the support vector machine [166], [185]. Although this requires that all samples be translated into the new space (in contrast, regular kernelized support vector machines only needs to know the kernel transformation and not transform every single sample), the overall performance is still better, leading to renewed attention for support vector machines.

So, going back to our server selection problem, to utilize support vector machines ideally we would have a set of training data with optimal offloading destinations already calculated previously. As mentioned before, this can still be done even without the knowledge of the optimal solution, by modifying the underlying objective function of the algorithm to a minimization of the service delay instead of a minimization of wrong classification, but both execution time and accuracy will be worse. Then, the feature space where the tasks will be plotted should be determined; this should reflect the important parameters of the tasks, such as its requirements and their physical origin. Then the kernelized support vector machine with stochastic gradient descent algorithm is applied once for each possible offloading destination, to determine the separator of tasks that are offloaded to that server and those that are not. For executing support vector machine at this stage, the tasks themselves have to be transformed through the kernel before applying stochastic gradient descent and ultimately finding the separators. Finally, with the separators calculated, the solution can be applied online to new tasks, where the requests are tested against the separators one by one until a positive match is found, at which point the task is offloaded to the corresponding server. Note how this requires a hierarchy of servers; how this priority is determined depends on the problem (e.g., servers with higher capacity may be tested first, or servers physically closer to the users). Thus, the ML algorithm only has to be trained once and can be used to quickly decide in online situations after training is done. The drawback here is that a good training set is needed so that it can be properly generalized later. See below the steps for modeling Mobile Edge Computing problems for support vector machines:

1) Determine what is the goal of the problem (e.g., maximizing the profit of the service provider).
2) Determine what will be classified by the algorithm (e.g., new incoming request from users).
3) Determine what are the possible classifications (e.g., rejected, allocated into one of the servers).
4) Identify what are the features of chosen data points (e.g., the requirements of the task).
5) Use feature selection techniques to remove unnecessary features, optimizing the algorithm.

In a more technical discussion, it is notable that the execution of the support vector machine is very simple, as testing a sample against the decision rule (which is determined offline) takes time proportional to the number of features. The main issue is determining the decision rule, which takes at worst-case time proportional to the number of samples, even with stochastic gradient descent. Thus, to make the execution smoother, it is important to select the most important features only. Once more, these complexities vary greatly

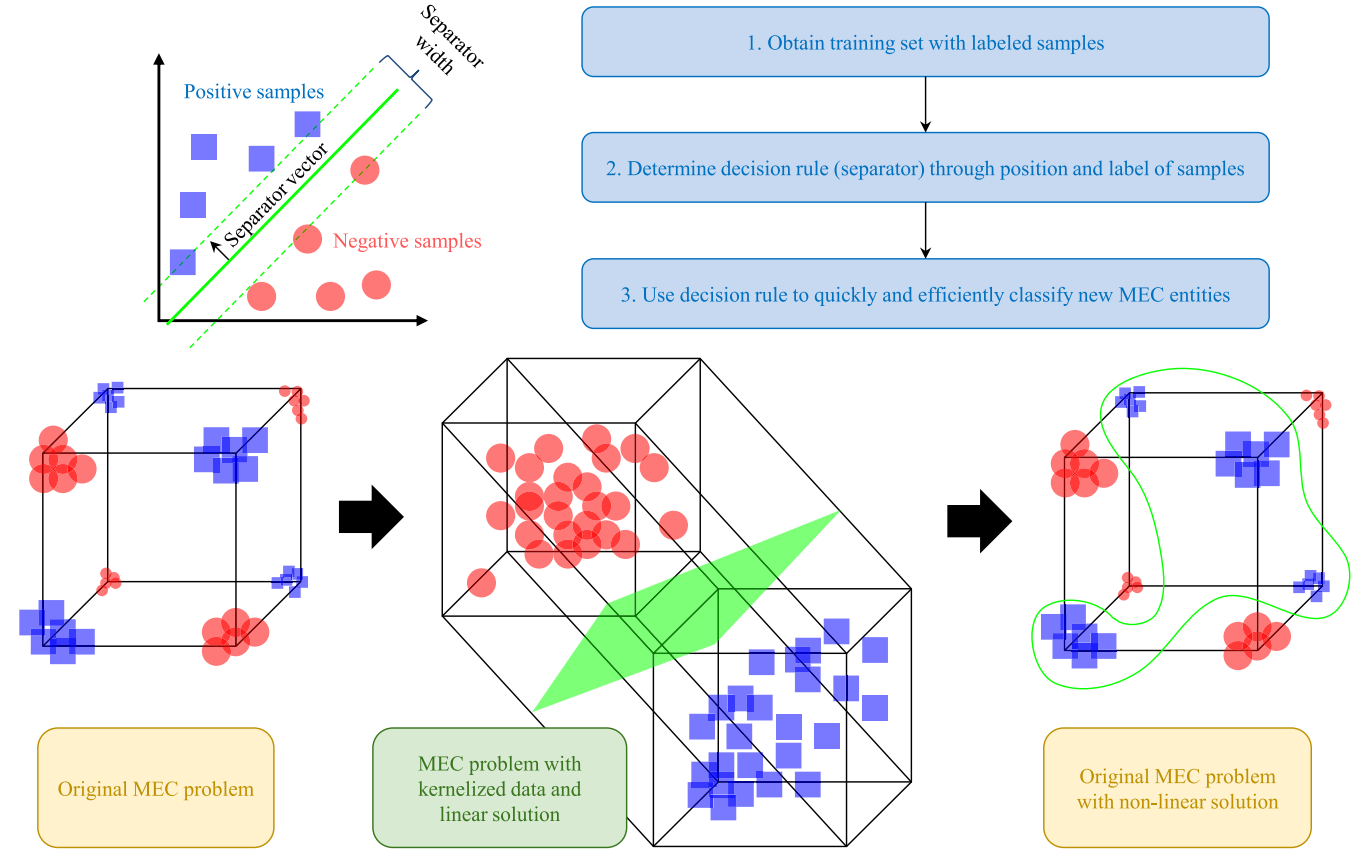

Fig. 6. Illustration of how support vector machines work, both with and without the kernel trick.

with problem and implementation, so these are rough estimates [186]. Regarding hyperparameters, the most pressing one is the choice of the kernel. There multiple well-established kernels to choose from, or it is also possible, albeit difficult, to design your transformation, customized for the problem in question and its data points. Ideally, many kernels should be tested looking for the best choice, but a good enough solution can be chosen instead if optimization is not needed by using one of the more popular kernels as is. Their performance is usually satisfactory [166], [183], [187].

Fig. 6 illustrates on top how the separator in support vector machines is determined. The separator is defined by a normal vector that is perpendicular to the separator, and some constraints that take into consideration all training data. All training samples are labeled as positive or negative regarding the classification. Moreover, the separator vector depends only on these training samples and Lagrange multipliers obtained from trying to maximize the width of the separator. By finding the vector, we have our decision rule for determining the classification of elements that were outside the training set. Finally, as illustrated in the bottom part of the figure, the algorithm can be applied even if the original problem does not have a linear solution by using a kernel to translate the scenario to different dimensions and finding a linear solution there.

The main advantage of support vector machines, especially the kernel variation, is to allow us to use a linear method to find a non-linear solution, which means a simple algorithm for a more complex problem. By adding stochastic gradient descent to the mix, the algorithm is capable of handling more data [184], [188]. All this make support vector machines a viable alternative for MEC problems, especially in how they are complex and usually come with massive amounts of data.

3) Bayesian Statistics: Another classic tool often used in ML algorithms is Bayesian statistics [189]. Given an independently and identically distributed variable whose particular distribution we do not know exactly, we can use Bayesian statistics to discover such distribution thusly: first, we can start by assuming the original distribution (such assumption can come from previous experience or expectations), which will be our prior; then, we sample the variable and, from the values we observe, we update our prior into a posterior. Through this method, the posterior is both affected by our previous experience in the form of the prior and our observations, meaning the model is capable of learning to become more accurate. This is called Bayesian inference. Some notable points: even if we have no guess/experience, we can use an uninformative prior, which is basically a generic distribution [190]; and, if a prior and its corresponding posterior have the same mathematical form, we are working with a conjugate prior [191]. Bayesian statistics offer helpful formulae for calculating the posterior based on data observed and the prior, which interested readers are recommended to investigate through the references in this subsection. These formulae are simple to calculate with conjugate priors, but still require to solve probabilistic integrations to calculate the evidence, a very complex feat which held back the usage of Bayesian statistics for a long time [192].

However, lately new techniques have gained popularity for at the very least approximating such integrals [193]–[196], making Bayesian statistics popular once again.

Bayesian inference alone is useful in many areas of research [197], [198], and MEC is no different. It is possible to utilize Bayesian inference for estimating the distribution of many variables. For example, we may want to know how often a user or a region of users generates tasks for allocating resources to that user/region; this is useful because we would not like to give too much or too little resources, as both situations are detrimental to the overall Quality of Service. Thus, through Bayesian inference, we can start with a guess of the distribution around the number of tasks per second and make this guess more and more accurate as we receive more tasks. Therefore, Bayesian inference is useful for receiving and creating information that is crucial for decision making. However, Bayesian statistics can also be used to directly make such decisions.

For that goal, we should look over to Bayesian networks [138], [199]. For using this technique, we first need a graph of our variables that represents the interdependencies among them as well. This is done by inserting a directed edge from variable *A* to variable *B* if the value of *A* affects the value of *B*. Given such a graph, Bayes' theorem [200] allows us to derive probability functions for all the nodes. For example, we can calculate the probability of $B = x$ given that $A = y$ and $C = z$. Moreover, just like in Bayesian inference, we can update these probabilities as we observe more and more data, giving us a learning mechanism. Just like before, readers are directed to the references in this subsection if they are interested in more mathematical details.

To see the full potential of Bayesian networks, let us walk through its usage for deciding offloading decision in Mobile Edge Computing. First, we must determine the dependencies, and what influences which server is the best choice. Some variables that definitely influence this would be the current load of each server, the latency between the user and the other servers, and the requirements (both communication and processing-wise) of the generated task. Then, given this graph, our next step will depend on what kind of learning we prefer. In the case of supervised learning, we need historical data that informs us what is the best offloading destination for multiple different scenarios; thus, given a good training set, our network can provide what is the probability of the optimal offloading destination being *x* if the scenario is $(a, b, c, \dots,)$. In the case of reinforcement learning, we will add a new variable to represent our objective function, e.g., minimize service delay. Then we will use our network to find out the probability of the service delay being minimum if the offloading destination is x and the scenario is $(a, b, c, \dots,)$. In both cases, the final product is the same: we use the Bayesian network to decide the offloading destination by finding the server with the highest probability of being the optimal destination (supervised learning) or with the highest probability of giving lowest delay (reinforcement learning). The main difference is that, with reinforcement learning, we can update the probabilities (and thus the decisions) as we observe the results of each offloading. Finally, it is important to note that, while it is possible to utilize continuous values, it does require a lot of data for the network to learn how to make a decision. Due to this, it is recommended to use more discrete values, using the labels "high service delay" or "low service delay" instead of specific values for example. For this, fuzzy logic [201], [202], another ML tool, could also be utilized. See below steps for modeling Mobile Edge Computing problems for Bayesian networks:

1) Determine what is the goal of the problem (e.g., maximizing the profit of the service provider).
2) Determine what will be decided by the algorithm (e.g., whether to reject incoming requests or allocate resources for them and how many resources to allocate).
3) Identify what features influence the decision (e.g., the requirements of the task).
4) Draw the interdependency graph of all variables and the decisions.
5) Use fuzzy logic or another technique for discretizing the values of the variables (optional).

Now, a more technical discussion. For both updating the Bayesian network with one sample and querying for the value desired, the complexity is proportional to the number of features, assuming that there is one variable per feature in the network. However, usually, there are extra nodes in the network to represent the desired metrics, which are not features. Nonetheless, the complexity should still be dominated by the number of features. Finally, note how this is just an estimate for the complexity, as the real value changes depending on the problem and implementation [203]. Regarding hyperparameters, there is obviously the decision between supervised and reinforcement learning, which ultimately should be taken based on how much historical data is available: if a lot of previous experience can be imparted onto the model, then supervised learning may be preferred; whereas, if not a lot of data exists, then reinforcement learning, despite its bad performance at the start, should be chosen for its better potential as more knowledge is acquired [151], [156]. Additionally, it is also necessary to choose between discrete and continuous values. Continuous values lead to more precise results but also need considerably more data. Discrete values are not so exact but lead to faster convergence with less training [204]. Note that it is possible to use continuous values for some variables and discrete values for others, and also to use more or less granularity for discrete values, so a lot of fine-tuning is possible if desired. As usual, determining the optimal hyperparameters requires experimentation and trials.

Fig. 7 shows on top how Bayesian inference is capable of closing on the real distribution of a variable and becomes more accurate as it is able to observe more samples. It is notable how in the beginning the output is very biased towards one side but it approaches the real distribution as more data is observed. At the bottom part of the figure we can see an illustration of how, from a Bayesian network graph, it is possible to derive the probability of each variable, which itself is a function of the influencing parameters (represented by incoming edges). By using such probabilities and Bayes' Theorem, it is possible to find the odds for every single scenario, based on observed samples. This is used to generate comparisons like the one on the right, where you evaluate which value for a

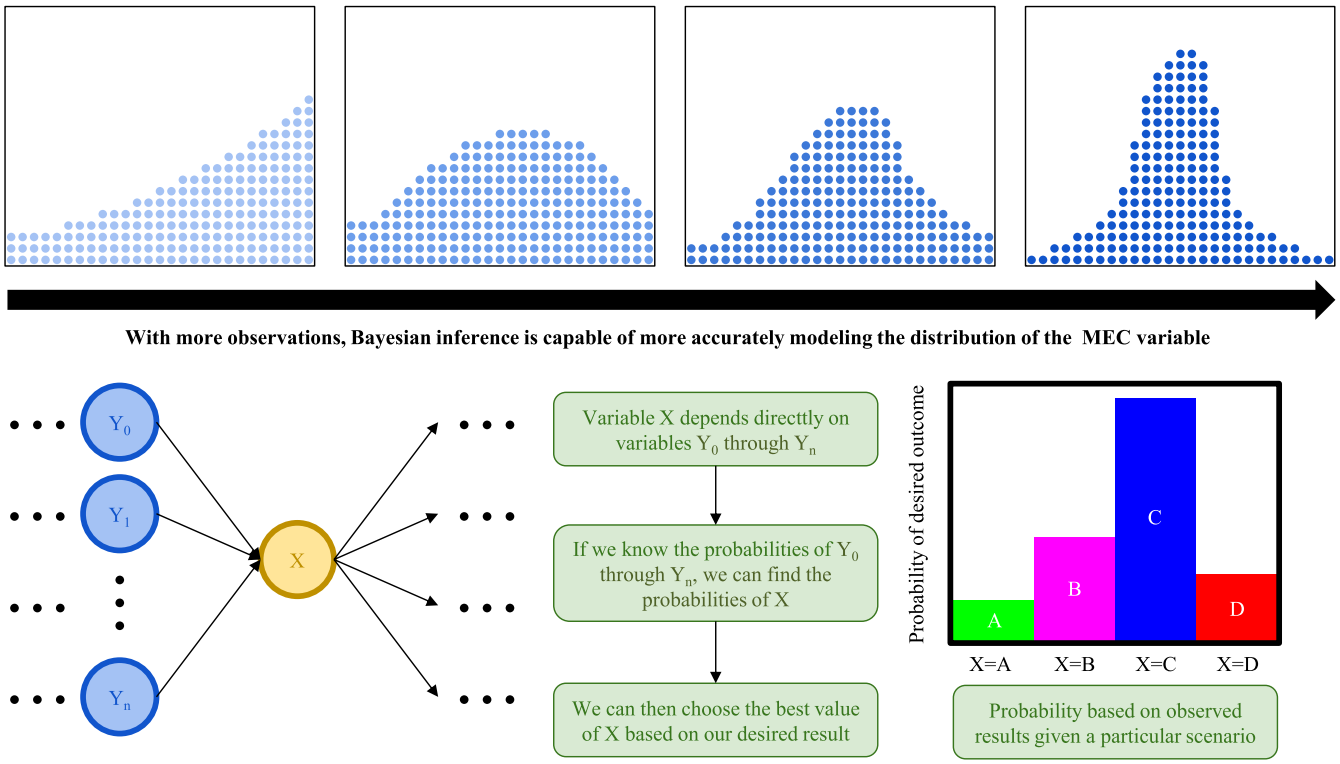

Fig. 7. Demonstration of use cases for both Bayesian inference and Bayesian networks.

single parameter (e.g., offload destination) results in a higher percentage of the desired result (e.g., low service delay) in the already seen cases, and use such information for decision making.

Bayesian inference is a very strong tool for MEC, especially for estimating unknown variables, as it can do that accurately. Moreover, Bayesian inference can take advantage of how MEC generates a lot of data (many users, many tasks, many servers, etc.) to make more accurate estimations. Similarly, Bayesian networks can also use this characteristic of MEC to be more accurate, offering better decisions. And, once it is trained, Bayesian networks are perfectly capable of producing efficient configurations quickly, which is a good fit for MEC, with its dynamic nature and ever-changing scenarios.

Lessons Learned: In this section, three advanced ML algorithms that have been trending on literature are presented. They are deep Q-learning neural networks, support vector machines with stochastic gradient descent and Bayesian statistics. These algorithms are not new, but recent advancements have improved them to the point of putting them in the forefront of ML application. Likewise, they can be applied to MEC as well. Deep Q-learning neural networks are excellent for decision making, even in complex situations with many parameters. Support vector machines are ideal for grouping data points and, with the kernel trick, they are capable of finding linear (i.e., quick and simple) solutions to even non-linear problems. Bayesian inference is a great tool for identifying the original distribution of variables and parameters, while Bayesian networks can be used for making decisions based on historical data and trying solutions that have worked consistently before. Moreover, these algorithms all are good for learning from data, which gives them the capability of improving. Bayesian statistics and deep Q-learning neural networks, in particular, can be set up as reinforcement learning, meaning they can improve while being used online and more quickly adapt to dynamic changes. All these characteristics allow the algorithms listed here to integrate well with MEC and their popularity recently is bound to lead to successful pieces of research.

IV. Machine Learning-Based Mobile Edge Computing

Previous sections introduced the fields of MEC and of ML. We also explained previously how ML is a perfect fit for the problems of MEC. This has been noticed by researchers and the literature demonstrates that. As such, this section will present existing works that use ML techniques to solve issues in the configuration of MEC systems. The references will be divided based on which MEC challenge they mainly tackle.

A. Offloading Decision

He *et al.* [205] use neural networks [164] in the context of smart cities to select which virtual network the user will connect to. In this context, the input is the user request, which must be classified into one of the pre-existing virtual networks. In their assumed scenario, a virtual network includes a base station (access point), a MEC server and a cache server. Thus, the algorithm must select which base station and which server to assign to the user, and whether or not the content requested should be cached. The choice is made so that the service provider's profit (the difference between operational cost and

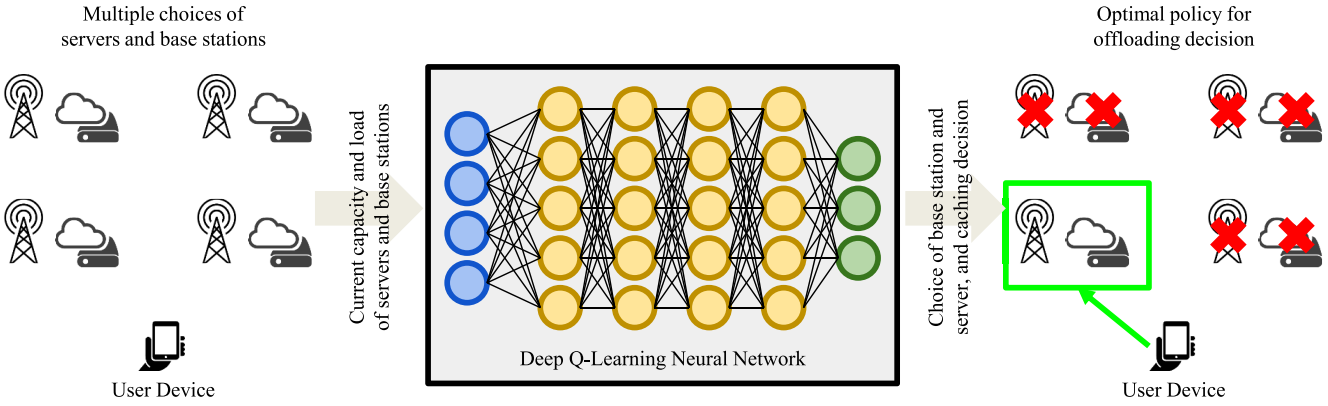

Fig. 8. How neural networks were utilized to decide the offloading destination for users in a smart city environment with MEC.

user revenue) is maximized. The neural network uses reinforcement learning (deep Q network [156]) to more quickly find the best selection based on the states of the servers and access points. The solution proposed by the authors is able to efficiently operate even in scenarios with multiple users and multiple servers, although they did ignore the potential of servers cooperating with each other. Moreover, the authors remark how neural networks, and consequently their proposed solution, are so adept at handling dynamic scenarios, with sudden changes, such as the ones in their envisioned MEC smart city. Fig. 8 illustrates how their decision process worked.

Rathore *et al.* [105] propose using a hesitant fuzzy set [206] to decide whether a user should offload to the MEC server or not based not only on the state of the resources (task execution time, server CPU usage and memory overhead) but also on the security level desired by the user. The performance is evaluated by a choice value function designed by the authors that takes delay and security into consideration. The authors justify their use of fuzzy logic by explaining how this technique is ideal for multi-variable requirements and specifications, such as the ones in their assumed scenario. Their approach is user-centric, with multiple services as alternatives to choose. Nonetheless, their proposed solution is shown to be able to choose the optimal service for the user.

Li *et al.* [207] chose to focus on the security aspect of a scenario involving IoT devices offloading to MEC servers. They assume that, after offloading to a server, the device learns about the current security risk of such server. This security data as well as the position of the device, both of which are variables that change with time, are used to train a multi-armed bandit framework [208], which is an ML algorithm. The framework is utilized for choosing which server to offload to. Additionally, the algorithm is based on reinforcement learning and thus can be trained online, as the device learns more about the servers. Finally, the proposal is further improved by allowing devices to share the knowledge gained by their models with other devices. Simulations based on synthetic and real data show that the proposal is capable of decreasing the number of successful jamming attacks when compared to a conventional method and of learning without cooperation between devices.

Sun *et al.* [101] offer interesting results in another offloading decision scenario for mobile users in a MEC system. Their scenario is highly dynamic, with moving users, varying wireless channel and even servers switching on and off. They formulate a problem where the tasks created by users must be executed before a deadline, with restrictions on the energy capacity of the user devices. The goal is to minimize the total delay. For achieving this, they utilize a variation of the multi-armed bandit algorithm [208]. Most notably, their proposal operates only with local knowledge, only having information of the user location, the task features (payload size and execution time), and the available MEC servers. Despite this, the proposal still behaves close to the same algorithm with knowledge of the state of all servers (how many resources they have allocated to other users and what is the access channel state related to them), and not too distant from a solution with full knowledge of the whole system, including future events. This demonstrates the potential of learning-based solutions, even with limited information.

Aral and Brandic [137] use tree-based naive Bayes [138] to select which MEC server and virtual machine a user should offload its tasks to with the goal of maximizing availability (i.e., time connected to the MEC system without any failure). This is accomplished by taking into account the historical data of previous failures for each server (physical and virtual) in order to train and let the Bayesian model learn. Moreover, their Bayesian network also receives information related to the user request, specifically the connection information (e.g., connection type and Internet service provider used). This combination of server and user information allows the Bayesian model to be more specific and more accurately provide the probability that each server will satisfy the requested availability. The authors point out that Bayesian networks are very adept in scenarios with dependent and independent probabilistic events, such as the ones in MEC. This is further shown through experiments that prove how Bayesian networks not only are better than conventional solutions at providing the availability required by users as they are better than other ML methods as well, namely Logistic Regression [139]. Fig. 9 illustrates the proposal of the authors. Note how the Bayesian network is used to relate events and calculate the probability of availability so that the server with the highest chance of success can be chosen. Fig. 9 illustrates the proposal of the authors. The Bayesian network in the center is only part of

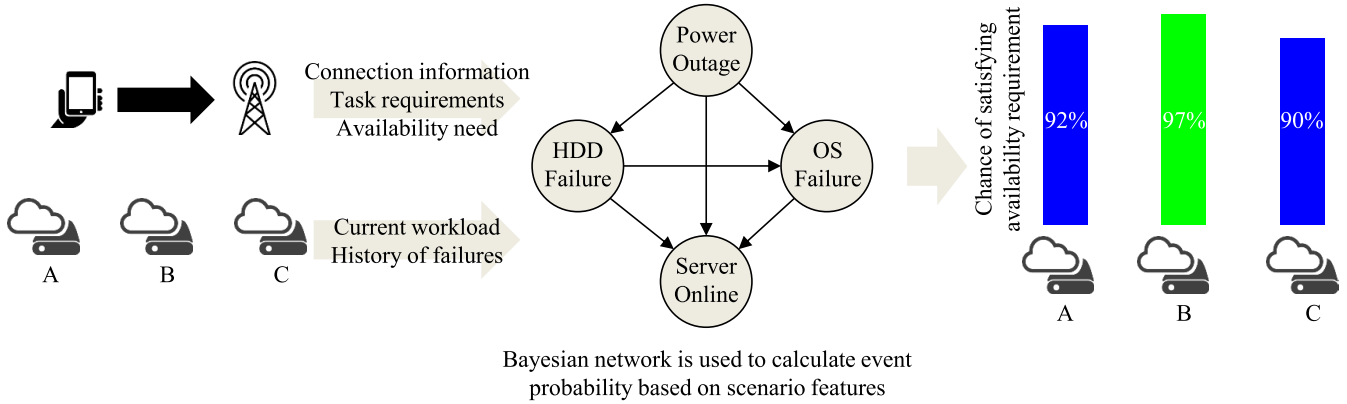

Fig. 9. Bayesian networks being used to calculate chance of server availability given the current scenario.

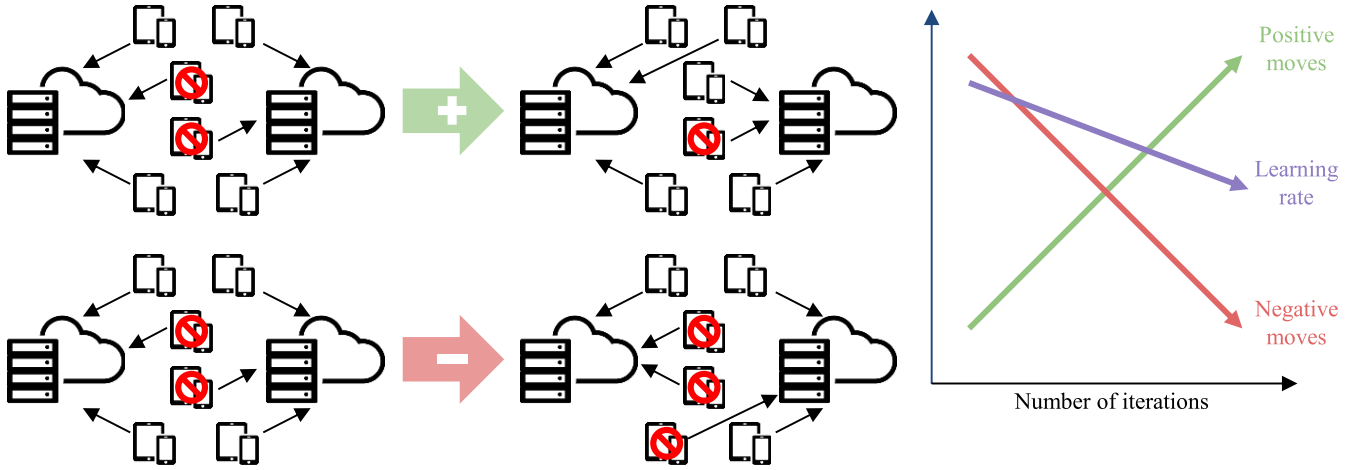

Fig. 10. Illustration of positive and negative moves in a Simulated Annealing solution and how the learning rate affects the ratio between those moves.

the full network in their proposal. Nonetheless, it serves as an example of the interdependency between events.

Carrega *et al.* [209] show their idea of a middleware for intermediating between a mobile device (such as smartphones or IoT devices) and a cloud system composed of edge and core servers. The middleware is responsible for deciding where each request of the end devices will be executed. For making such a decision, their framework utilizes Simulated Annealing [148] as well. The authors implemented their framework and experiments show not only its feasibility but also a better performance than a conventional solution used for deciding offloading destination.

Kim *et al.* [76] offer an algorithm for deciding the offloading target for MEC users. They first start by implementing a convex solution to the problem that is able of finding the optimal configuration. However, they remark that such a solution is not feasible because its complexity scales too quickly with the number of users and servers. Moreover, the convex algorithm has to be executed again after changes in the scenario (e.g., arrival of a new user, user mobility). Thus, they also present a practical solution based on the ML algorithm Simulated Annealing [148]. As shown in Fig. 10, their algorithm considers works by making swaps involving users with bad quality of service. The swapped users trade connected servers. A positive move decreases the number of users with low quality while negative moves increase it. The problem is that only accepting positive moves usually leads to local optimums and misses global optimums. Simulated Annealing works around this by having a variable learning rate that decreases with each iteration. Thus, the algorithm has a high chance of accepting a negative move in the beginning, but this decreases with time. Additionally, negative moves that decrease the fitness too much have lower chances of being accepted, even at the beginning of the execution. It is notable how their proposal has the caveat that users will always utilize the edge server associated to their nearest access point if its capacity permits; the algorithm is only executed when such server is full and a neighboring server must be chosen, in which case the proposal will determine the best server for this. Nonetheless, the learning-based solution is proven through simulations to achieve higher server efficiency and service more users when compared to a greedy algorithm and a method without cooperation between edge servers.

B. Resource Allocation

Rodrigues *et al.* [21] use Particle Swarm Optimization [144] to decide, in the fronthaul, the transmission power of base stations with an associated MEC server in order to minimize service delay for users while considering the transmission

between static users and the base stations and the execution time of the user-generated tasks. Since in their assumed scenario the offloading decision is also associated with the transmission power of the servers, the ML solution is also used for balancing the workload across the servers. The solution is proven to provide near-optimal levels of service delay with a much lower execution time when compared to a greedy algorithm.

Rodrigues *et al.* [67] also use Particle Swarm Optimization [144] in a MEC scenario with user mobility (the previous work considered static users) where the algorithm once again decides in the fronthaul the base station transmission power. Moreover, in this work, the objective is to maximize the system capacity regarding users that can be serviced with a maximum delay requirement. Their solution also determines virtual server migration, including the routing and bandwidth for such action. The authors prove that ML is able to utilize the MEC servers more efficiently and serve more users than other, non-learning-based solutions.

Zhang *et al.* [90] propose an algorithm that utilizes a dynamic evolutionary game [145] in a MEC system in a heterogeneous network to allocate in the fronthaul communication and computation resources to users. The objective is to minimize a cost function that involves service delay, provider profit, and energy consumption. The algorithm decides how much bandwidth and processing time each mobile terminal (which is their end device) receives from the MEC server such that the resource capacities of the server are respected. The proposal is shown to provide both better energy consumption and service delay than conventional methods from the literature, despite working with high numbers of servers and users.

Wang *et al.* [146] utilize a heuristic solution that combines both a genetic algorithm [147] and simulated annealing [148] in a MEC scenario with cooperation with the remote cloud. Notably, the algorithm decides how many resources, i.e., caching memory and processing capacity, the remote cloud server will allocate in the backhaul to each cloudlet, to execute the tasks that those cloudlets cannot handle, in order to minimize service delay. Thus, there is cooperation between the multiple edge servers and the single remote, more powerful server. However, the edge servers cannot share workloads between themselves. The proposal is shown to be better than a conventional solution with remote storage and local execution, as well as better than approaches that are based solely on genetic algorithms or solely on simulated annealing.

Wen *et al.* [108] propose a method for allocating resources to users with a data stream application. Since solving the resource allocation problem for all users would be too complex and result in too much overhead according to them, they first divide the users randomly into small groups. A genetic algorithm [147] adapted by the authors is then utilized for three goals: allocating servers for each group (for the execution of the applications from the users), allocating bandwidth for each group (so the users can communicate with the servers), and for distributing the resources inside each group. Additionally, the algorithm is capable of allocating resources from both edge servers and a remote server, with corresponding capacities and different latencies. The algorithm is also responsible for deciding, for each user, whether their application will be executed locally or remotely, through the cloud. While it is normal to assume that dividing the users into groups randomly would decrease the performance of the solution, the authors show through simulations that not only the genetic algorithm achieves a near-optimal performance regardless, its throughput is also very close to the one achieved by a genetic algorithm without grouping, but with a significantly smaller execution time.

Xu *et al.* [149] attempt to address the difficulty of deploying MEC outside of city centers. They remark that such a scenario will involve scarce power supply for the edge servers and must depend on green energy, which is unstable. Thus, their proposal is based on optimizing not only the service delay, to provide high quality for the users, but also the operational cost especially in terms of consumed energy. To achieve this, they model the scenario as a Markov decision process and apply both Q-learning [150] and post-decision state learning [149] to decide how many resources and which server(s) are allocated to the user. Moreover, their solution is capable of scaling such allocations as the requests from the user changes. Simulations show that the proposal fares better when compared to a solution that ignores previous states and historic data (i.e., no learning) and a solution without post-decision state learning (i.e., only Q-learning).

Yang *et al.* [151] tackle the problem of allocating communication and computation resources from a single MEC server to multiple users. They explain how the latency and the reliability of the communication are affected not only by the bandwidth given to each user but also by the coding block length assigned, which itself affects the decoding error probability. This is conventionally done analytically, but the problem can easily become intractable due to the stochastic nature of MEC. Thus, the authors opt to model the scenario as a Markov decision process and utilize deep Q-learning [152] for deciding how much processing power and transmission resources are given to each user at each time frame. Compared to benchmark methods that ignore the state of the system, the proposal achieves a higher task success ratio.

C. Server Deployment

Crutcher *et al.* [94] use a k-nearest neighbors algorithm [210] to cluster tasks based on their data size. Then, MEC servers are assigned to each server according to the amount of bandwidth available and the state of the server processors. The goal of this work is to minimize both service delay and energy consumption. The assumed scenario is user-centric, with mobile users and the possibility to utilize multiple servers. Nonetheless, the servers do not communicate between themselves. Interestingly, the authors train their ML model with only partial knowledge of the network state and user profile, but their solution is still capable of predicting the status of the system and making efficient choices.

Hou *et al.* [140] propose a caching strategy in MEC servers where the offered content is clustered based on its access

features using k-means clustering [141] and transfer learning [211]. The goal here is to correlate contents with similar access profiles, put them in the same cluster and then deploy virtual MEC servers to cache themin order to maximize cache hit rate and lower transmission delay. The transfer learning allows the models to be trained locally and then shared and combined between servers. This turns out to be key, as the proposed solution is shown through simulations to be better than a learning technique without server cooperation, as well as better than random caching and caching only of more recently used content.

Li *et al.* [96] also utilize k-means clustering [141], but with the goal now being to decide where the MEC servers should be deployed by clustering users based primarily on their physical location but also considering the processing requirement of their tasks. In their assumed scenario, servers must be deployed in the same site as a pre-existing base station. A MEC server is deployed for each cluster of users, with the final location being decided based on the base stations associated to the users of that cluster and the ensuing network delay between the base station and the MEC server, all while trying to minimize the overall service delay. The learning-based solution is shown to more efficiently utilize the servers than a conventional greedy technique.

Du *et al.* [142] propose a variation of Clustering by Fast Search and Find of Density Peaks (CFSFDP) [143] for clustering together similar user-generated tasks in order to deploy for each cluster a single MEC server, which in turn can be specialized for that type of task and thus result into a more efficient service. Their algorithm is designed so that any features could be used, such as communication payload size and computation time requirement. Their proposal is shown to perform more efficiently than other, non-learning-based solutions.

In a similar way, Liu *et al.* [212] use k-means clustering [141] to cluster users based on their features and associate each cluster with the same MEC server with the goal of having specialized servers that are more efficient in their corresponding tasks. Since their proposal is done in a smart campus environment, they cluster students into study-oriented, active-type, and enclosed-type based on their behavior and applications used. This clustering is performed mainly based on the semantic data associated with the users. The proposal is capable of offering a high-quality service despite the high number of students connected to the system.

Li and Wang [97] also tackle the server placement problem. The deployment of the servers is done trying to simultaneously minimize the energy consumption and keep the service delay under a threshold, through a multi-objective optimization problem. Servers can be placed only in association with existing, pre-deployed base stations. The authors utilize Particle Swarm Optimization [144] to decide where each server will be deployed and which base stations will offload the tasks they receive to it. The algorithm must respect the delay threshold, respect the capacity of each server in terms of processing tasks, and minimize the energy consumption of the whole system. The authors model energy as a base consumption for servers that receives an increment proportional to the task workload of the servers. Obviously, such objective function, with multiple goals and constraints, is not simple. Nonetheless, the ML solution not only is feasible regarding execution time, but it also provides lower energy consumption and higher resource efficiency than random and greedy placement policies. Moreover, the learning-based algorithm is able to find an efficient configuration even with a low number of iterations.

D. Overhead Management

Gu *et al.* [134] use a fuzzy control model [135] for deciding whether subtasks (represented in their work by Java classes) should be executed locally or in a MEC server. This decision is based on the available resources in the user device, the state of the wireless network between both agents and the characteristics of the subtask (i.e., resources required and whether it can even be executed remotely) and attempts to minimize the overall delay. The final result is an interesting algorithm that is capable of deciding what to offload and when to do it based on the current status of the system.

He *et al.* [106] model a security requirement issue in MEC through a constrained Markov decision process [136], where the goal is to choose whether to execute a task locally or in the MEC server based on the system state (wireless channel state and amount of tasks in MEC buffer) such that energy consumption and delay are minimized without violating a minimum privacy level required by the user. The scenario is however fairly simple, being user-centric and only considering one MEC server to choose from besides the option of local execution. Nonetheless, simulations evidence that the Markov decision process solution is better than a conventional scheme at preserving privacy.

Dinh *et al.* [153] also tackle the challenge of deciding where users should offload their requests, but now in a multi-user multi-server MEC scenario where users can decide to which server they will offload to and what is their transmission power when doing so. Higher transmission power means more subtasks reaching the server, but also result in higher energy consumption. Users may also choose to process tasks locally, but that also requires energy. The authors show that analytical methods are feasible for this problem, but only if a static scenario is considered. In a dynamic situation, where the wireless channel state keeps varying (a much more realistic assumption), a learning-based solution is needed, which is why they utilize a model-free Q-learning algorithm [154]. Simulations show that their proposal is capable of achieving near-optimal performance regarding consumed energy and throughput, but with feasible execution time.

Tan and Hu [155] work with vehicles as end devices in a MEC system. The authors remark how the mobility of vehicles and how quick the scenario changes are challenges that can only be satisfactorily addressed through ML. Thus, they propose utilizing deep Q-networks [156] for determining both offloading policies and caching strategies. In their scenario, there are edge servers installed as roadside units and also other vehicles can be utilized both as caching servers or as targets for task offloading. Thus, their proposed algorithm decides both where a task should be offloaded to and, in the case of content request, whether such content should be cached and

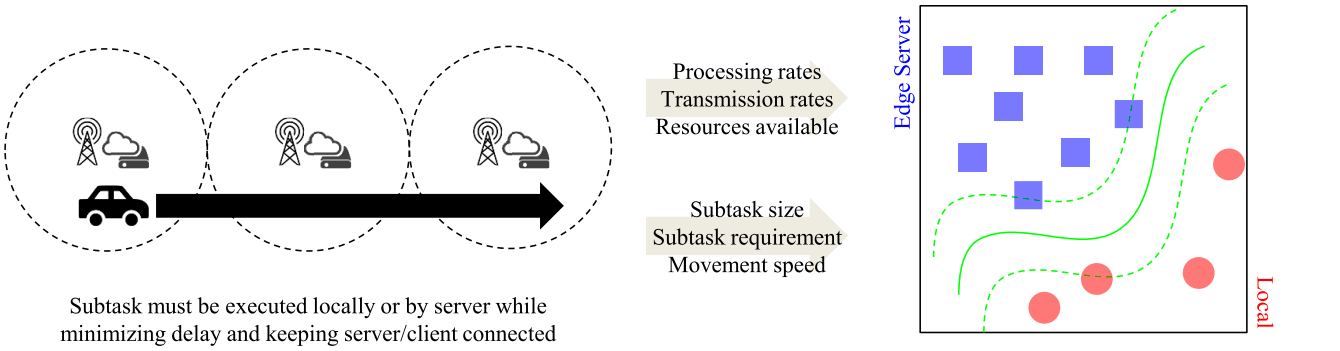

Fig. 11. In a scenario with vehicular users, how support vector machines can be used to decide whether subtasks should be offloaded or not.

where. Their solution is able to do this even amidst the dynamicity of both vehicular networks and MEC networks, and it is even able to take into account the movement of the vehicles into the states of the deep Q-network.

Vita *et al.* [181] present a scenario with mobile users being serviced by a MEC network with multiple edge servers and a remote core server. As the users move around, it is highly likely that what was once a suitable host server for their corresponding virtual machine now presents a low-quality service. Thus, the authors propose an algorithm for deciding when to migrate virtual machines and data between servers, including the one at the core of the network. The algorithm is a deep Q-learning neural network [152] that takes into consideration the position of the users as they move as well as the state of each server and how many users they are already hosting. The authors remark that their solution has a competitive performance while requiring fewer data and time to execute.

Gao *et al.* [157] similarly propose a method for deciding either migration of virtual servers or caching of requested content. These authors consider a scenario with mobile users and a set of edge servers and remote servers. They design an algorithm that, given the user trajectory and the state of each of the servers, decides whether data should be moved preemptively or not, where data could be either the virtual server associated to that user or content that user may request. Such algorithm utilizes both Ant Colony Optimization [158] and multi-feature linear discriminant analysis [159], both learning-based algorithms. The authors compare their proposal with other conventional methods for preemptively caching content and with a method based on only utilizing mobile cloud computing. The proposal is shown to provide lower delay even with high workloads.

Finally, Wu *et al.* [160] use Support Vector Machines [161] in a scenario with MEC servicing users in vehicles. The authors say that vehicles usually move too quickly for MEC servers to finish the user tasks in time. Thus, their proposal is used to classify the subtasks of the user into two groups: local execution or server execution. If a subtask is chosen for server execution, it means it can be executed and its output returned before the user leaves the reach of the base station associated with that server. The support vector machine is trained based on features that are key for determining if the subtask can be completed: the local and server processing rates, the task size, number of resources (i.e., memory) required and available, and the transmission rate. The learning model is trained so that the service delay resulting from its classification is minimized. The authors show that their proposal achieves significant improvement when compared to local only execution and random offloading. Furthermore, the authors also show that support vector machine has very low execution time when compared to other learning algorithms and, depending on the kernel chosen, can be only 3% off the optimal value. Fig. 11 illustrates the proposal of the authors. Although not shown in the figure, the non-linear solution (represented by the non-linear separator) is obtained by utilizing the kernel trick, explained in the previous section.

Lessons Learned: This section shows a list of recent and relevant pieces of literature that utilize ML solution to address MEC problems. Each research work is accompanied by a corresponding description of its contents. It is possible to see some patterns in most of these research works that could be taken as guidelines for using ML with MEC. First, many of the references start by providing a convex, non-learning based solution to their chosen problem and proving how that solution is not scalable if the scenario has too many users, servers, applications, etc., further corroborating our conclusion that better solutions, like ML, are needed in this area. Second, almost all of the works listed here show that ML leads to better performance than conventional solutions, especially regarding overhead and complexity. Finally, there are some patterns regarding the usage of algorithms and MEC problems. Namely, ML classification algorithms (e.g., neural networks and their variations) are often used for offloading decision problems since they can choose the best server among the existing options; ML optimization algorithms (e.g., Particle Swarm Optimization, multi-armed bandit) are used for resource allocation since they are better at finding solution for continuous problems; ML clustering algorithms (e.g., k-means clustering) are usually used for server deployment problems, finding clusters of users to determine where to put the servers; finally, algorithms with state transitions and set of actions (e.g., Q-learning) are fine choices for overhead management in that they can choose actions that balance the overhead cost and the performance improvement. This information is summarized in

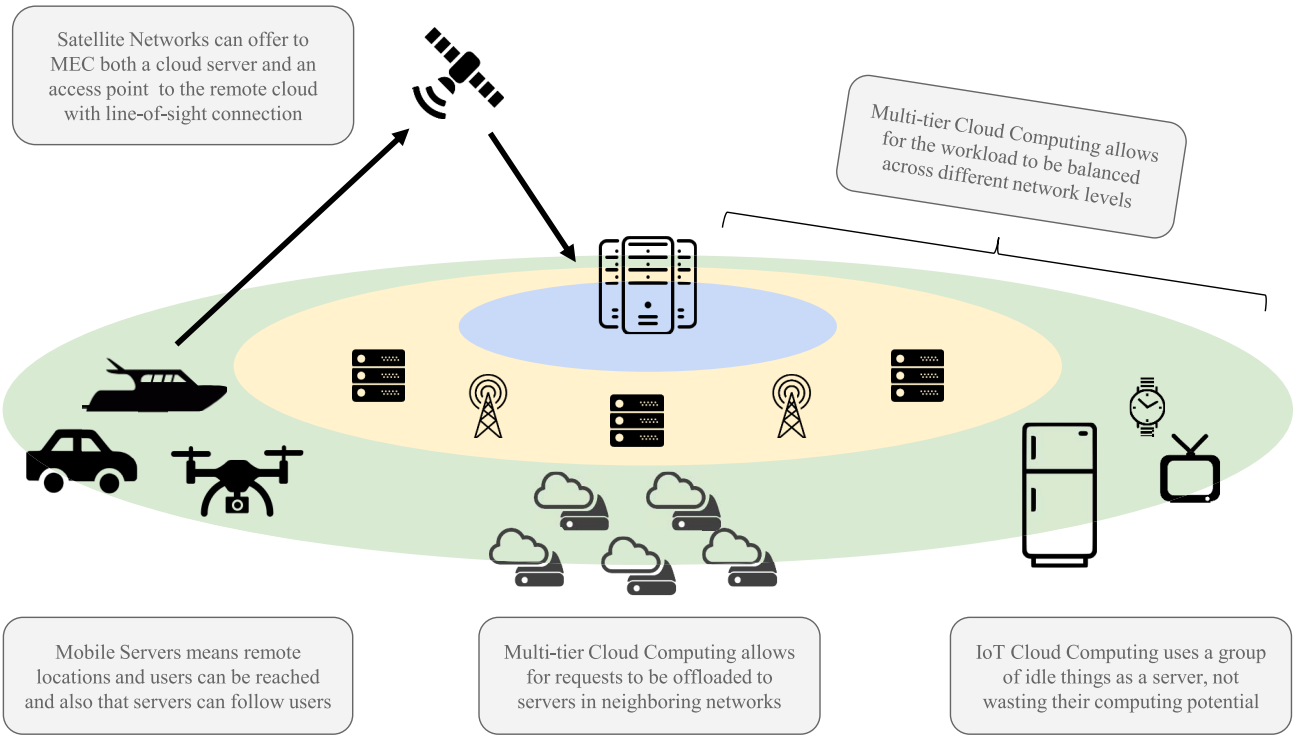

Fig. 12. Future research directions and MEC applications that open up with the usage of ML in this type of service model.

TABLE IV
WHAT TYPE OF ML SOLUTION IS USUALLY USED FOR EACH CLASS OF MEC CHALLENGE

MEC Challenge	ML Category	Algorithm Example
Offloading decison	Classification	Deep Neural Network [164]
Resource allocation	Optimization	Simulated Annealing [148]
Server deployment	Clustering	k-Means Clustering [141]
Overhead management	State transitions	Q-learning [150]

Table IV. Not all references follow this and the best advice is to use these recommendations as a starting point. The contents of this section should help researchers to find out what has already been researched in the field as well as justify utilizing ML in MEC.

V. FUTURE PERSPECTIVES

ML is a very important tool currently for MEC research, as shown in the previous sections. However, on top of that, these learning algorithms open up a lot of new possibilities for MEC applications. ML is capable of enabling the usage of new technologies and deployment strategies that can enhance the quality of the service given to the users as well as the profit of the service providers. This section will present and discuss some of these areas that have a big potential for research works and it will also expose how ML is able to integrate into each one of these ideas. Fig. 12 shows a visual representation of the future perspectives presented in this segment.

A. Mobile Servers

Existing MEC research assumes a system with fixed servers [27], [28], [60], and for good reason. The deployment is simpler this way, as it is easier to install edge servers in buildings and base stations, with existing communication infrastructure and energy supply. Moreover, the configuration is also easier, as mobile servers would add the complexity of managing their movement efficiently. Nonetheless, there are relevant benefits to allow the servers to move. First of all, not all regions and places, especially the ones not in urban areas, have the infrastructure to deploy a fixed server [149], [213]. Mobile servers would enable MEC to reach these areas and the users therein. Additionally, it is important to take into considerations that even in urban areas, there is a certain migration of users as people go from their homes to work and then back. Mobile servers are capable of following this migration and offer true ubiquitous access and computing by moving the servers from the city centers to the suburbs together with the clients. In order to achieve the same with fixed servers, it would be necessary to deploy more servers, near both the homes and the workplaces of the users, which is an expensive solution; or subject the clients to a longer latency in at least one of those areas by deploying all servers in the other location; or, as a final option, use as many servers as the mobile solution but spread them thinly across the suburbs and the centers, leading to a risk of overload. There are also other secondary use cases, such as providing MEC to sensors in remote areas like forests or deserts, or even in post-disaster areas, to connect the victims in disaster areas and provide them with some form of infrastructure. All of these are possible research areas within MEC with mobile servers that have received little to no attention in the literature, offering many opportunities for future work.

It is important to mention here that mobile servers have been exploited in other areas. For example MANETs [214], GPRS [215], and streaming services [216]. This proves

that mobile servers are a valid option for expanding MEC. Moreover, while it is true that a lot of research has been done to solve problems related to server mobility, the solution found in other fields do not necessarily automatically apply to MEC. Cloud offloading has its peculiarities, mostly related to task submission, result retrieval, and the virtual expansion of the virtual environment [20], [21], not to mention the scale of users and servers that should arise in the future. Because of these characteristics, it is important to invest research efforts in at least adapting existing mobile server solutions from other fields to MEC. Indeed, the special features of MEC are what makes it so solutions used in other network research areas (from MANETs and streaming services to the highly similar conventional cloud computing) do not apply to MEC [27]. Both with or without mobile servers, specific research for MEC is needed.

Mobile MEC can be realized in some different ways. For example, servers can be mounted on land vehicles, like cars or vans, giving more availability in land areas and allowing the servers to be driven to remote areas where there is no MEC infrastructure. The obvious drawback here is that the vehicles need roads or something similar at the very least to reach such locations. Another possibility is to mount servers on sea vehicles, such as boats or ships, allowing MEC with mobile servers to be utilized in coastal areas even in places far from roads that cannot be reached by land vehicles. However, this limits the reach of MEC to places near bodies of water. One other option is to mount servers on unmanned aerial vehicles (UAVs) [217]. UAVs do not need any roads, sea or river, so they are ideal for hard to reach places and increase the potential of MEC deployment. However, one important limitation is the weight capacity of UAVs [218], which means not all types of servers can be used in this kind of system. Moreover, all these vehicles have fuel limitations that impose a maximum on how far they can travel. Nonetheless, these multiple types of vehicles and their corresponding advantages and disadvantages bring variation in potential research and create even more opportunities.

The main difficulty for this area must be the management and coordination of the travel routes. Besides all the other variables involving servers, such as load and resource allocation, now it is also needed to decide how they will move. For this, it is important to consider the density of users and to try to minimize the amount of handover between servers, which would lead to higher delay and lower quality of service. It is also desirable to keep a server near the user at least until the completion of the current task, otherwise it is needed to coordinate the delivery of the results through a relay between the servers, adding more overhead and delay. Obviously, all these are incremental challenges in an area with enough complexity as is. Indeed, determining the best routes and movements in problems like this is usually non-polynomial and dealt with using ML even without the added difficulty of delivering MEC service [219]. Thus, it is safe to assume that a good course of action for achieving MEC with mobile servers is using ML techniques.

For example, it is possible to model the scenario as a state transition machine, where the movement of the servers are the possible actions, and let deep Q-learning neural networks learn the best movement patterns depending on the resulting performance of each route. Alternatively, Bayesian networks can be used to select routes with a higher percentage of the desired outcome, as defined by a preferred objective. Finally, there are other ML algorithms not described in this paper that would be ideal for route selection jobs like the ones in mobile server MEC, such as Ant Colony Optimization [158]. These solutions can be applied to mobile server scenarios with different objectives, such as lower delay or higher provider profit, opening up a lot of exciting possibilities for the area of MEC and in research.

B. Multi-Tier Cloud Computing

Most research on MEC found in the literature works with limited amounts of servers. In fact, there is a substantial number of works that consider a single server [13], [93], [108]. Others consider a limited amount of servers in an area of interest [18], [74], [94]. At most, we can see works that offer collaboration between the edge and a remote cloud [77], [85], [220], which is useful because it allows for requests with high leniency for latency to be offloaded to the remote cloud while the edge is reserved for requests that demand low latency. However, real-life networks operate with more than two tiers. There are local area networks, wide area networks, metropolitan area networks; there are femtocells, microcells, macrocells; we have the backbone of the network as well as access networks/the edge of the network. At all of these points, we can potentially deploy cloud servers that can operate in conjunction with MEC. Of course, to manage this convoluted system is no simple task, since the load of various servers has to be considered as well as very different latencies needed to access each one. To complicate matters further, it is unrealistic to assume that all servers are homogeneous; it makes more sense to deploy less powerful servers in smaller networks and deploy servers with higher capabilities in higher tiers. However, on the other hand, heterogeneous servers are harder to work with, especially for conventional algorithms, since it is necessary to track the different capacities of each one.

Nonetheless, moving away from the hardships, it is clear that multi-tier cloud computing, especially involving MEC, comes with sizeable benefits [84]. As mentioned before, requests can be assigned to different tiers depending on how much latency they require, or even with how much processing time or payload data they have associated with them. Tasks that demand more communication or need shorter latency can be assigned to the edge or closer tiers, while requests that need more computation can be sent to farther away servers that are more powerful. Additionally, with multiple tiers, it is possible to divide the tasks in more than the simple two groups of low latency/high latency. This higher granularity allows for a more precise assignment and offloading, resulting in higher service quality. Additionally, in situations where local networks and its resident servers are overloaded, it is possible to offload some of its requests to servers in neighboring cells, especially if the servers there happen to not have many requests [76].

This balance of the workload results in a more efficient use of the resources and higher service quality.

The main reason that research with many servers is so scarce is also one of the main strengths of this concept: a high variety of choices. If there are more options of servers to connect and offload to, this means that choosing one for each user/request is now more complicated, as more features and parameters must be taken into consideration. In fact, even if you consider only MEC, without multi-tiers, a greedy approach for each server selection is already difficult enough. A viable alternative is to group users/requests depending on their characteristics, and ML is proficient at this. Both support vector machines and Bayesian networks can classify the input into groups and then assign them to different tiers. Support vector machines would realize such task by first learning from a training set of users with their corresponding optimal servers and then classifying new requests later. Meanwhile, Bayesian networks are capable of learning online, by measuring how its choices of servers perform and which tiers have a higher probability of achieving high-quality service based on the tasks' features. Additionally, even deep Q-learning networks are applicable, by modeling the scenario as a state transition machine with request arrivals and server selection as the main action. Finally, other ML grouping techniques, such as k-Means Clustering [141], [221], could also be used.

Once again, multi-tier cloud computing is a very promising area for the future, as it is the closest representation of real-life networks and allows full use of the resources deployed. As such, research in this field is very important for the advancement of cloud computing. Moreover, there are multiple approaches to explore in multi-tier cloud computing, with various objectives to choose and exploration both vertically and horizontally in the tiers. Nonetheless, independently of the preference, ML grouping is likely the most viable approach for managing this kind of system.

C. Satellite Networks

One field of study that has garnered a lot of attention recently is satellite communications [222]. Using a satellite as a connection point has the benefit of a higher chance of establishing line of sight, as the satellite is in a prime position and the antenna for communicating with the satellite is also presumably positioned carefully with this in mind. However, there is also the significant drawback of latency, as the distance between users and the satellite is considerable, and the satellite itself has to communicate to servers on the ground besides the user, to forward the queries it receives. That, together with sending back the reply of such queries, means going through the distance between ground and space four times, which adds up in term of latency. A possible workaround of this is deploying the servers directly onto the satellites, effectively halving the latency as the communication is now exclusively between the user and the satellite only. This is particularly effective with cloud computing, as cloud servers are capable of serving many of the requests from the users. In this fashion, satellite servers can be used as an aid to MEC, where real-time requests are served on the edge while the satellite handles other requests to guarantee that the edge servers are not overloaded and can respond quickly as we need it.

The biggest point holding back the inclusion of satellites as servers with MEC is basically the price involved in sending servers to space [222]. Indeed, it is a big investment to ask of cloud service providers. Thus, it is more common to see proposals involving the use of satellites only as communication relays between the users and the remote cloud infrastructures [222]–[225]. Thus, there are two possible approaches for using satellites in this environment: either utilize them as a communication infrastructure but configure the system so that delay is lowered as much as possible; or deploy servers at the satellites but improve profit and decrease cost by, for example, increasing efficiency for the cloud resources. Both objectives can be achieved through clever server selection and allocation, making sure task requirements are sufficiently satisfied, but not overly so. The goal is to simultaneously accept as many requests as possible (increasing profit) and taking advantage of the edge servers to satisfy latency intolerant requests (softening the impact of the distance to the satellites).

Server selection could be done through conventional algorithms, but ML is ideal for this situation due to the execution time of learning-based solutions. After proper training, ML can select the appropriate server quickly for the new requests, lowering the overhead cost while still efficiently using the system. Thus, a good method for this would once again use ML grouping, to separate requests between edge, satellite and remote cloud (i.e., beyond the satellite) based on the features of the tasks and the status of the servers. That is a considerable amount of data to go through for making a decision, but ML excels in such scenarios. Thus, deep Q-learning neural networks can be used, where the action is which server the task will be offloaded to. Alternatively, support vector machines can be used for grouping as normal, or Bayesian networks can be utilized to select the server that has a higher probability of giving low overall delay/high profit given the task profile and previous performances.

Thus, MEC with satellite support is a very interesting topic for the future, somewhat impeded by the cost of such deployment and the long latencies related to it. But, while this is a difficult challenge, solving it is bound to bring significant benefits for the area, so it is definitely a recommended research direction. A very important tool to achieve this is, of course, ML algorithms.

D. Offloading to IoT Devices

In the future with IoT, there will be many devices connected to the network all around us. These will range from home appliances to vehicles and wearable pieces of equipment. These devices all have processing capabilities that stay idle for a significant amount of time since they are not necessarily being used at all times. This idle processing potential could be taken advantage of by cloud systems. For example, tasks created by users could be divided into subtasks that are sent separately to different devices. This way, even though each individual device is relevantly inferior in capability to a full server, by using many of them simultaneously it is possible

to efficiently offload requests from users. This can be done in conjunction with conventional edge and remote cloud servers to create a system with more resources. This would necessitate, of course, some type of lightweight interface for the IoT devices to receive these subtasks and process them that does not interfere with their main functionalities. The payback is significant since it adds a lot of resources from the IoT to the cloud network to be used by the clients. As a matter of fact, the potential is so big that it has already attracted the attention of the literature [226].

However, the big complication is in the maintenance of such a system. Adding IoT devices as potential servers raises the number of possible destinations (and therefore choices to be made) by a lot. Moreover, these devices are not servers and they have their own jobs that will contend for the local resources but with higher priority than the subtasks from the cloud system. This also must be taken into consideration when offloading subtasks to them, trying to avoid sending requests to IoT devices that are currently busy lest the service quality will decay. Moreover, when dividing tasks into subtasks, more likely than not those subtasks will be interdependent, so they may have to wait for one another or be processed at the same place [227], [228]. These are complications to an already complex decision that is choosing among a myriad of servers. Since multiple server problems are already scarce in the literature, it is reasonable to expect that conventional solutions are also far from ideal in this envisioned IoT-as-a-server network.

Given the massive amount of devices in IoT, it is not possible to use ML grouping with each device as a possible classification when deciding where to offload requests. A more viable alternative may be to separate this into two processes. First, the decision of whether to process the tasks locally, at an edge server, at a remote server or in an IoT device is made. Then, if the algorithm chooses to offload to an IoT device, a second algorithm is employed to decide which device. Moreover, when choosing between IoT devices only, a pre-screening can be employed to remove all devices that are already busy, with either IoT tasks or cloud tasks. This would significantly reduce the number of choices, which conversely improves the efficiency of the ML algorithms.

Thus, for choosing where (locally, edge, remotely, IoT, etc.) the request will be executed, deep Q-learning neural networks (the possible actions for the state transition are offloading to each category), support vector machines (each category is a different group for classification) or Bayesian networks (based on previous performances, calculate the chance for each category of delivery the desired result) can be used. Then, if necessary, for choosing which IoT device to offload to, from the algorithms in this paper, the most suitable is Bayesian network, since they are able of giving the chance of success for each possible device and it is easier for us to simply ignore the devices that are not available. Alternatively, deep Q-learning neural networks can also be used by adding an extra feature that identifies the availability of the device. Either way, ML is better capable of choosing the best server in these scenarios with many choices, especially compared to conventional, non-learning solutions.

Tapping into the potential of idle IoT devices would be a great addition to cloud systems and could help enhance even further the performance of MEC by giving more resources to the edge. There is a lot of potential research to be done on this field, trying to find ways to efficiently use the IoT resources and properly integrate them with MEC infrastructures. Furthermore, this would unite two areas that have attracted a lot of interest and investment from industry and academia recently: cloud computing and IoT. Nonetheless, managing such a massive system is only viable through ML, since learning algorithms are more efficient at making choices when so many options are available.

E. Dynamic Predictions

MEC operates under very dynamic conditions. Users are capable of moving, which means possibly moving away from base stations, connecting to new access points, changing the latency between them and the servers. Obviously, this itself can overload access networks quickly [66], [67], if there is mass migration of users, which not only can lead to collision but, if the requests from those users are sent to the edge servers associated with the corresponding base station, then we will also encounter a situation where the servers themselves can suddenly and quickly become overwhelmed. Beyond mobility, it is also possible to have scenarios where there are sudden local peaks of requests sent by the users (e.g., events with big crowds, experiments being performed in universities or companies). If too many requests suddenly arrive at the system, chances are that the current configuration and offloading policies are not appropriate and service quality will tank. And that is not all: MEC equipment may fail, with servers and/or access points becoming unavailable suddenly and for a while. Obviously, when your configuration was built under the assumption that all existing resources are usable, losing some of them will require changes, which once again means a decay at service quality.

Ignoring the threat of these dynamic and unnexpected changes is not advisable, as client dissatisfaction resulting from these events can be devastating for the marketability of the system [73]. However, as mentioned, such incidents are sudden, with little warning. Conventional solutions to this basically amount to providing redundancy [77], allocating backup resources to be deployed in case of technical failures or sudden overloads, but this is expensive and means having a lot of your assets idle, which is obviously a waste in case there is no dynamic change and may even be financially unviable in the first place. A better alternative is to predict the dynamic changes and make the appropriate modifications beforehand (e.g., reserving extra resources right before a peak in usage, making a backup of user data right before a server failure) [82]. There are plenty of ML methods for predicting events based on historical data and the current status of the system, so this is another situation where Artificial Intelligence can be efficiently applied.

For example, Bayesian networks can be used to calculate the probability of dynamic incidents, informing us what is the chance of a server failure or a server/access point becoming

overloaded based on previous observations. We could also use Bayesian inference to derive the distribution of user mobility and task output and use such information to infer the load in the system. Moreover, we could also use deep Q-learning neural network in a model where actions are the events happening in the system, such as user mobility or equipment failure, and the objective function is based on how accurate the neural network can predict these events. Both these techniques are based on reinforcement learning, so not only they can learn from event logs but also they will improve their predictions the longer they are utilized.

The other subsections are different from dynamic prediction because the techniques suggested there could possibly be implemented without ML (even if this is recommended for efficiency-related reasons). Predicting events in MEC, however, is something almost surely enabled only by ML, which is arguably the best method for identifying patterns in data [44]. Research based on this would be especially useful for MEC service providers in the industry, allowing them to offer a more stable service to their users, which is bound to improve their position. Thus, there should be no shortage of interest or investment for this kind of work.

F. ML Integration

As a final point, it is also important, when integrating MEC and ML, to consider how ML will be executed inside the system. Usually, ML research presumes full knowledge of the system for the algorithm to take advantage of [21], [85], [88], but this data is produced across many different places and points. For example, let's say you want your neural network to decide the offloading destinations. For this, you need to know the status of the edge servers and it would also be helpful to have the rate of requests coming from the users as well as the requirements of each of those requests. However, servers in the edge are located apart from each other and the same is true for users. If you want your algorithm to have all that information to perform decisions, then you must design a system for centralizing such data.

Obviously, it would take time for the servers to deliver such reports to a centralized office, negating at least partially the biggest advantage of MEC over the conventional cloud. It is possible for the servers to perform these decisions based on local information only. For example, you could execute deep Q-learning neural networks, support vector machines and even Bayesian networks based solely on the data available on a single server, training the models using the load of that server only and the tasks that are sent directly to it. This means that other servers would not have to report their observations, cutting down the overhead time of the algorithm. Moreover, there are techniques for knowledge transfer in ML, where models are trained on local information and then the model themselves are combined, meaning less data has to be transferred [110], [140]. However, the drawback is that partial knowledge offers less potential than full knowledge, meaning that solutions and configurations from models trained with full knowledge will usually result in higher quality service, regardless of the objective function [220].

Thus, there is a tradeoff between the overhead of centralizing the full system knowledge and the quality degradation from operating with partial knowledge. It is possible to operate within clusters of the network, sending the knowledge to one server that is not central to the whole network but still obtaining more data about the system than a single local edge server. Regardless, this is a complicated decision that necessitates careful research, comparing the actual overhead cost of centralizing all the information in the system with how much data the algorithms need to produce efficient configurations (i.e., how much partial knowledge is enough). There probably is not a single overarching answer for all ML algorithms and MEC problems regarding this question, meaning a lot of work in order to create realistic learning solutions but also plenty of opportunity for research. Finally, it is also worthy to point out although this knowledge integration problem is indeed a complication for ML, other convex and conventional MEC solutions that do not implement learning must also contend with this for their algorithms.

Lessons Learned: This section presents future possibilities for MEC research. The ideas shown here integrate existing technology with MEC in order to improve cloud computing, not necessarily limiting ourselves to the edge but always with the goal of improving service. Namely, the proposed ideas here are: mobile servers, balancing workload horizontally and vertically across different network levels and tiers, offloading to groups of idles devices in Internet of Things, using satellites as access points and/or servers, dynamically predicting the changes that MEC scenarios go through, and how to properly integrate the ML algorithms. All these ideas have great potentials that are mostly untouched in the literature. Moreover, ML solutions, such as the ones presented in the previous section, are key to realizing the ideas proposed here, especially because most of them increase the number of choices and decisions to be made, necessitating a method for making quick and efficient solutions, something that ML is adept at.

VI. Conclusion

MEC is important for future network designs, such as 5G and Internet of Things, for enabling mobile devices and applications with low delay threshold, increasing the range of what can be executed in mobile devices by connecting them to cloudlet servers in the edge of the network. However, the efficiency of MEC is dependent on challenges that have high dimensionality, such as deciding where to offload the tasks generated by users and how many resources to allocate to each connection. Because these challenges have too many variables and a high dimensionality solution space, conventional methods, such as heuristic algorithms and convex optimization, either simplify the scenario too much or are not efficient enough to solve the challenge. Thus, we propose the utilization of ML, a paradigm of algorithms that allow computers to learn the intricacies of each problem in other to efficiently survey the solution space and find optimal (or near-optimal) solutions in a quick way. Indeed, ML is such a good fit for MEC, that several pieces of research in the literature utilize an ML algorithm for efficiently operating MEC, which are

properly listed in this survey. These works even corroborate the appropriateness of the ML category/MEC challenge pairs we formulated. Furthermore, ML is capable of increasing the limits of what can be achieved through MEC, allowing more complex systems to be built. Such systems, in turn, come with the benefit of more resources and a better chance of high-quality service. For example, MEC with ML can enable the usage of mobile servers, satellite servers, more cooperation between servers in different networks, and even using idle IoT devices like servers and predicting the state of the network. Indeed, these are all areas with high payoff potential that are in need of research.

We sincerely hope that this survey encourages future works at utilizing ML for MEC operation since we believe this combination will lead to efficient networks and cloud services in the future, especially because we believe the potential of ML algorithms has not been fully realized in the area of MEC and that the current amount of research involving both fields is not enough when the true benefits of this combination is considered. For this end, we present suggestions of which ML algorithms to use for MEC, giving details on how to adapt them to MEC in general but, more importantly, to perspectives for the future of MEC. This way, we believe this present survey can be a guide for future research.

ACKNOWLEDGMENT

The research results have been achieved by "Research and Development on Intellectual ICT System for Disaster Response and Recovery", the Commissioned Research of National Institute of Information and Communications Technology (NICT), JAPAN.

REFERENCES

[1] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities," in *Proc. 10th IEEE Int. Conf. High Perform. Comput. Commun.*, Sep. 2008, pp. 5–13.

[2] Q. Ding, B. Tang, P. Manden, and J. Ren, "A learning-based cost management system for cloud computing," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 362–367.

[3] L. Columbus. (Oct. 2017). *Cloud Computing Market Projected to Reach 411B by 2020*. Accessed: Mar. 1, 2018. [Online]. Available: https://www.forbes.com/sites/louiscolumbus/2017/10/18/cloud-computing-market-projected-to-reach-411b-by-2020/#370bda7278f2

[4] *Google Cloud Platform: Google Cloud Computing, Hosting Services and API*. Accessed: Mar. 1, 2018. [Online]. Available: https://cloud.google.com/

[5] *Amazon Web Services (AWS)—Cloud Computing Services*. Accessed: Mar. 1, 2018. [Online]. Available: https://aws.amazon.com/

[6] *Microsoft Azure Cloud Computing Platform and Services*. Accessed: Mar. 1, 2018. [Online]. Available: https://azure.microsoft.com/

[7] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[8] S. Al-Janabi, I. Al-Shourbaji, M. Shojafar, and M. Abdelhag, "Mobile cloud computing: Challenges and future research directions," in *Proc. 10th Int. Conf. Develop e-Syst. Eng. (DeSE)*, Jun. 2017, pp. 62–67.

[9] M. Satyanarayanan, "Fundamental challenges in mobile computing," in *Proc. 15th Annu. ACM Symp. Principles Distrib. Comput.*, May 1996, pp. 1–7.

[10] M. López-Nores, Y. Blanco-Fernández, J. J. Pazos-Arias, A. Gil-Solla, and M. Ramos-Cabrer, "Augmented reality, smart codes and cloud computing for personalized interactive advertising on billboards," in *Proc. 10th Int. Workshop Semantic Soc. Media Adapt. Pers. (SMAP)*, Nov. 2015, pp. 1–6.

[11] A. A. Ateya, A. Vybornova, R. Kirichek, and A. Koucheryavy, "Multilevel cloud based tactile Internet system," in *Proc. 19th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2017, pp. 105–110.

[12] G. Premsankar, M. D. Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[13] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[14] R. Saadoon, R. Sakat, and M. Abbod, "Small cell deployment for data only transmission assisted by mobile edge computing functionality," in *Proc. 6th Int. Conf. Future Gener. Commun. Technol. (FGCT)*, Aug. 2017, pp. 1–6.

[15] A. A. Laghari, H. He, M. Shafiq, and A. Khan, "Assessing effect of cloud distance on end user's quality of experience (QoE)," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Oct. 2016, pp. 500–505.

[16] K. Fogarty. (Mar. 2013). *In the Cloud, Distance Matters for Compute Efficiency*. Accessed: Mar. 2, 2018. [Online]. Available: https://www.networkcomputing.com/data-centers/cloud-distance-matters-compute-efficiency/1889266701

[17] K. Bierzynski, A. Escobar, and M. Eberl, "Cloud, fog and edge: Cooperation for the future?" in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, May 2017, pp. 62–67.

[18] H. Huang, Y. Cai, and H. Yu, "Distributed-neuron-network based machine learning on smart-gateway network towards real-time indoor data analytics," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, Mar. 2016, pp. 720–725.

[19] L. Wang, O. Brun, and E. Gelenbe, "Adaptive workload distribution for local and remote clouds," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Oct. 2016, pp. 3984–3988.

[20] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[21] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "A PSO model with VM migration and transmission power control for low service delay in the multiple cloudlets ECC scenario," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[22] M. Aazam and E.-N. Huh, "Fog computing and smart gateway based communication for cloud of things," in *Proc. Int. Conf. Future Internet Things Cloud*, Aug. 2014, pp. 464–470.

[23] S. Lavanya, N. M. S. Kumar, S. Thilagam, and S. Sinduja, "Fog computing based radio access network in 5G wireless communications," in *Proc. Int. Conf. Wireless Commun. Signal Process. Netw. (WiSPNET)*, Mar. 2017, pp. 559–563.

[24] *IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing*, IEEE Standard 1934-2018, pp. 1–176, Aug. 2018.

[25] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[26] ETS Institute. (2019). *Multi-Access Edge Computing (MEC)*. Accessed: May 15, 2019. [Online]. Available: https://www.etsi.org/technologies/multi-access-edge-computing

[27] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.–Dec. 2009.

[28] Z. Zhang and W. Hao, "Development of a new cloudlet content caching algorithm based on Web mining," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 329–335.

[29] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[30] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018.

[31] V. Balasubramanian, N. Kouvelas, K. Chandra, R. V. Prasad, A. G. Voyiatzis, and W. Liu, "A unified architecture for integrating energy harvesting IoT devices with the mobile edge cloud," in *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, Feb. 2018, pp. 13–18.

[32] *5G: The Internet for Everyone and Everything*. Accessed: Mar. 2, 2018. [Online]. Available: http://www.ni.com/pdf/company/en/Trend_Watch_5G.pdf

[33] *Internet of Things Forecast*. Accessed: Mar. 2, 2018. [Online]. Available: https://www.ericsson.com/en/mobility-report/internet-of-things-forecast

[34] E. G. Renart, D. Balouek-Thomert, X. Hu, J. Gong, and M. Parashar, "Online decision-making using edge resources for content-driven stream processing," in *Proc. IEEE 13th Int. Conf. e-Sci. (e-Sci.)*, Oct. 2017, pp. 384–392.

[35] K. Intharawijitr, K. Iida, H. Koga, and K. Yamaoka, "Practical enhancement and evaluation of a low-latency network model using mobile edge computing," in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2017, pp. 567–574.

[36] B. Ahlgren, M. Hidell, and E. C.-H. Ngai, "Internet of Things for smart cities: Interoperability and open data," *IEEE Internet Comput.*, vol. 20, no. 6, pp. 52–56, Nov./Dec. 2016.

[37] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley, "A brief survey of machine learning methods and their sensor and IoT applications," in *Proc. 8th Int. Conf. Inf. Intell. Syst. Appl. (IISA)*, Aug. 2017, pp. 1–8.

[38] S. Das and M. J. Nene, "A survey on types of machine learning techniques in intrusion prevention systems," in *Proc. Int. Conf. Wireless Commun. Signal Process. Netw. (WiSPNET)*, Mar. 2017, pp. 2296–2299.

[39] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM J. Res. Develop.*, vol. 3, no. 3, pp. 210–229, Jul. 1959.

[40] J. R. Koza, F. H. Bennet, III, D. Andre, and M. A. Keane, "Automated design of both the topology and sizing of analog electrical circuits using genetic programming," in *Artificial Intelligence in Design*, J. S. Gero and F. Sudweeks, Eds. Dordrecht, The Netherlands: Springer, 1996.

[41] C. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.

[42] N. M. Nasrabadi, "Pattern recognition and machine learning," *J. Electron. Imag.*, vol. 16, no. 4, pp. 1–2, Oct. 2007.

[43] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, no. 2, pp. 95–99, Oct. 1988.

[44] O. Simeone. (Aug. 2018). *A Very Brief Introduction to Machine Learning With Applications to Communication Systems*. Accessed: May 9, 2019. [Online]. Available: https://arxiv.org/abs/1808.02342

[45] Z. M. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432–2455, 4th Quart., 2017.

[46] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self-organizing cellular networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2392–2431, 4th Quart., 2017.

[47] R. Mishra, P. Verma, and R. Kumar, "Gateway discovery in MANET using machine learning and soft computing: A survey," in *Proc. Int. Conf. Innov. Inf. Embedded Commun. Syst. (ICIIECS)*, Mar. 2017, pp. 1–6.

[48] N. Z. B. Zubir, A. F. Ramli, and H. Basarudin, "Optimization of wireless sensor networks MAC protocols using machine learning: A survey," in *Proc. Int. Conf. Eng. Technol. Technopreneurship (ICE2T)*, Sep. 2017, pp. 1–5.

[49] M. A. Alsheikh, S. Lin, D. Niyato, and H. P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1996–2018, 4th Quart., 2014.

[50] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud," *J. Netw. Comput. Appl.*, vol. 82, pp. 93–113, Mar. 2017.

[51] M. Demirci, "A survey of machine learning applications for energy-efficient resource management in cloud computing environments," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2015, pp. 1185–1190.

[52] X. Fei *et al.*, "CPS data streams analytics based on machine learning for cloud and fog computing: A survey," *Future Gener. Comput. Syst.*, vol. 90, pp. 435–450, Jan. 2019.

[53] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[54] W. Yu *et al.*, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[55] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[56] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2359–2391, 4th Quart., 2017.

[57] M. Kaur, *A Comprehensive Survey on Architecture for Big Data Processing in Mobile Edge Computing Environments: From Hype to Reality*. Cham, Switzerland: Springer Int., 2019.

[58] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2586–2595, Nov. 2017.

[59] G. Yang, Q. Sun, A. Zhou, S. Wang, and J. Li, "Access point ranking for cloudlet placement in edge computing environment," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2016, pp. 85–86.

[60] L. Zhao, W. Sun, Y. Shi, and J. Liu, "Optimal placement of cloudlets for access delay minimization in SDN-based Internet of Things networks," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1334–1344, Apr. 2018.

[61] C. Meurisch *et al.*, "Temporal coverage analysis of router-based cloudlets using human mobility patterns," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[62] S. Kächele, C. Spann, F. J. Hauck, and J. Domaschka, "Beyond IaaS and PaaS: An extended cloud taxonomy for computation, storage and networking," in *Proc. IEEE/ACM 6th Int. Conf. Utility Cloud Comput.*, Dec. 2013, pp. 75–82.

[63] G. Liu, "Research on independent SaaS platform," in *Proc. 2nd IEEE Int. Conf. Inf. Manag. Eng.*, Apr. 2010, pp. 110–113.

[64] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs containerization to support PaaS," in *Proc. IEEE Int. Conf. Cloud Eng.*, Mar. 2014, pp. 610–614.

[65] M. Kozlovszky, M. Törőcsik, T. Schubert, and V. Póserné, "IaaS type cloud infrastructure assessment and monitoring," in *Proc. 36th Int. Convent. Inf. Commun. Technol. Electron. Microelectron. (MIPRO)*, May 2013, pp. 249–252.

[66] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.

[67] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1287–1300, Sep. 2018.

[68] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.

[69] Q. Fan and N. Ansari, "Workload allocation in hierarchical cloudlet networks," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 820–823, Apr. 2018.

[70] L. Zhao and J. Liu, "Optimal placement of virtual machines for supporting multiple applications in mobile edge networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6533–6545, Jul. 2018.

[71] L. Chen, S. Patel, H. Shen, and Z. Zhou, "Profiling and understanding virtualization overhead in cloud," in *Proc. 44th Int. Conf. Parallel Process.*, Sep. 2015, pp. 31–40.

[72] N. Tziritas *et al.*, "Data replication and virtual machine migrations to mitigate network overhead in edge computing systems," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 4, pp. 320–332, Oct. 2017.

[73] I. Farris, T. Taleb, M. Bagaa, and H. Flick, "Optimizing service replication for mobile delay-sensitive applications in 5G edge network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[74] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[75] C. Liu *et al.*, "A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure," *IEEE Trans. Services Comput.*, vol. 11, no. 2, pp. 249–261, Mar./Apr. 2018.

[76] S.-Y. Kim, X. de Foy, and A. Reznik, "Practical service allocation in mobile edge computing systems," in *Proc. 27th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Nov. 2017, pp. 1–6.

[77] B. P. Rimal, D. P. Van, and M. Maier, "Mobile-edge computing versus centralized cloud computing over a converged FiWi access network," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 498–513, Sep. 2017.

[78] A. Kapsalis, P. Kasnesis, I. S. Venieris, D. I. Kaklamani, and C. Z. Patrikakis, "A cooperative fog approach for effective workload balancing," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 36–45, Mar./Apr. 2017.

[79] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, Sep. 2014.

[80] D. Zhao, M. Mohamed, and H. Ludwig, "Locality-aware scheduling for containers in cloud computing," *IEEE Trans. Cloud Comput.*, to be published.

[81] D. F. Towsley, "Mobility models for wireless networks: Challenges, pitfalls, and successes," in *Proc. 22nd Workshop Principles Adv. Distrib. Simulat.*, Jun. 2008, p. 3.

[82] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proc. IEEE 27th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2016, pp. 1–6.

[83] A. N. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 398–401, Jun. 2017.

[84] A. Kiani and N. Ansari, "Toward hierarchical mobile edge computing: An auction-based profit maximization approach," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2082–2091, Dec. 2017.

[85] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Commun. Lett.*, vol. 21, no. 7, pp. 1481–1484, Jul. 2017.

[86] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu, "Efficient resource allocation for on-demand mobile-edge cloud computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8769–8780, Sep. 2018.

[87] C. Vicentini, A. Santin, E. Viegas, and V. Abreu, "A machine learning auditing model for detection of multi-tenancy issues within tenant domain," in *Proc. 18th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput. (CCGRID)*, May 2018, pp. 543–552.

[88] T. G. Rodrigues *et al.*, "Towards a low-delay edge cloud computing through a combined communication and computation approach," in *Proc. IEEE 84th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2016, pp. 1–5.

[89] R. Tandon and O. Simeone, "Harnessing cloud and edge synergies: Toward an information theory of fog radio access networks," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 44–50, Aug. 2016.

[90] J. Zhang *et al.*, "An evolutionary game for joint wireless and cloud resource allocation in mobile edge computing," in *Proc. 9th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2017, pp. 1–6.

[91] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[92] D. Ganguly, M. H. Mofrad, T. Znati, R. Melhem, and J. R. Lange, "Harvesting underutilized resources to improve responsiveness and tolerance to crash and silent faults for data-intensive applications," in *Proc. IEEE 10th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2017, pp. 536–543.

[93] X. Lyu *et al.*, "Optimal schedule of mobile edge computing for Internet of Things using partial information," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Nov. 2017.

[94] A. Crutcher, C. Koch, K. Coleman, J. Patman, F. Esposito, and P. Calyam, "Hyperprofile-based computation offloading for mobile edge networks," in *Proc. IEEE 14th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2017, pp. 525–529.

[95] K. Han *et al.*, "Application-driven end-to-end slicing: When wireless network virtualization orchestrates with NFV-based mobile edge computing," *IEEE Access*, vol. 6, pp. 26567–26577, 2018.

[96] B. Li, K. Wang, D. Xue, and Y. Pei, "K-means based edge server deployment algorithm for edge computing environments," in *Proc. IEEE SmartWorld Ubiquitous Intell. Comput. Adv. Trusted Comput. Scalable Comput. Commun. Cloud Big Data Comput. Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Oct. 2018, pp. 1169–1174.

[97] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 66–73.

[98] K. Xiao, Z. Gao, Q. Wang, and Y. Yang, "A heuristic algorithm based on resource requirements forecasting for server placement in edge computing," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 354–355.

[99] M. Marjanović, A. Antonić, and I. P. Žarko, "Edge computing architecture for mobile crowdsensing," *IEEE Access*, vol. 6, pp. 10662–10674, 2018.

[100] L. Tianze, W. Muqing, Z. Min, and L. Wenxing, "An overhead-optimizing task scheduling strategy for ad-hoc based mobile edge computing," *IEEE Access*, vol. 5, pp. 5609–5622, 2017.

[101] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.

[102] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23511–23528, 2018.

[103] H. Trinh *et al.*, "Energy-aware mobile edge computing and routing for low-latency visual data processing," *IEEE Trans. Multimedia*, vol. 20, no. 10, pp. 2562–2577, Oct. 2018.

[104] H. Zhang, Z. Chen, J. Wu, and K. Liu, "FRRF: A fuzzy reasoning routing-forwarding algorithm using mobile device similarity in mobile edge computing-based opportunistic mobile social networks," *IEEE Access*, vol. 7, pp. 35874–35889, 2019.

[105] S. Rathore, P. K. Sharma, A. K. Sangaiah, and J. J. Park, "A hesitant fuzzy based security approach for fog and mobile-edge computing," *IEEE Access*, vol. 6, pp. 688–701, 2018.

[106] X. He, J. Liu, R. Jin, and H. Dai, "Privacy-aware offloading in mobile-edge computing," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Singapore, Dec. 2017, pp. 1–6.

[107] S. Matoussi, I. Fajjari, S. Costanzo, N. Aitsaadi, and R. Langar, "A user centric virtual network function orchestration for agile 5G Cloud-RAN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–7.

[108] H. Wen, L. Yang, and Z. Wang, "ParGen: A parallel method for partitioning data stream applications in mobile edge computing," *IEEE Access*, vol. 6, pp. 5037–5048, 2018.

[109] D. Li, B. Dong, E. Wang, and M. Zhu, "A study on flat and hierarchical system deployment for edge computing," in *Proc. IEEE 9th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Las Vegas, NV, USA, Jan. 2019, pp. 163–169.

[110] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient machine learning in 2 KB RAM for the Internet of Things," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, Aug. 2017, pp. 1935–1944.

[111] A. Mestres *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 3, pp. 2–10, Jul. 2017.

[112] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer-Verlag, 2009.

[113] M. Kang and N. J. Jameson, "Machine learning: Fundamentals," in *Prognostics and Health Management of Electronics: Fundamentals, Machine Learning, and the Internet of Things*. Hoboken, NJ, USA: Wiley, 2019.

[114] J. G. March, "Exploration and exploitation in organizational learning," *Org. Sci.*, vol. 2, no. 1, pp. 71–87, Feb. 1991.

[115] C. Luo, S. Kumar, and D. N. Mallick, "Achieving excellence in exploration and exploitation by matching appropriate resources," *IEEE Eng. Manag. Rev.*, vol. 46, no. 3, pp. 103–107, Sep. 2018.

[116] I. AlQerm and B. Shihada, "Enhanced machine learning scheme for energy efficient resource allocation in 5G heterogeneous cloud radio access networks," in *Proc. IEEE 28th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, Montreal, QC, Canada, Oct. 2017, pp. 1–7.

[117] J. Li, Z. Zhao, and R. Li, "Machine learning-based IDS for software-defined 5G network," *IET Netw.*, vol. 7, no. 2, pp. 53–60, Mar. 2018.

[118] J. S. Perez, S. K. Jayaweera, and S. Lane, "Machine learning aided cognitive RAT selection for 5G heterogeneous networks," in *Proc. IEEE Int. Black Sea Conf. Commun. Netw. (BlackSeaCom)*, Jun. 2017, pp. 1–5.

[119] M. Chen, W. Saad, and C. Yin, "Liquid state machine learning for resource allocation in a network of cache-enabled LTE-U UAVs," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Singapore, Dec. 2017, pp. 1–6.

[120] Q. Yao *et al.*, "Core, mode, and spectrum assignment based on machine learning in space division multiplexing elastic optical networks," *IEEE Access*, vol. 6, pp. 15898–15907, 2018.

[121] E. Balevi and R. D. Gitlin, "Unsupervised machine learning in 5G networks for low latency communications," in *Proc. IEEE 36th Int. Perform. Comput. Commun. Conf. (IPCCC)*, San Diego, CA, USA, Dec. 2017, pp. 1–2.

[122] J. Riihijarvi and P. Mahonen, "Machine learning for performance prediction in mobile cellular networks," *IEEE Comput. Intell. Mag.*, vol. 13, no. 1, pp. 51–60, Feb. 2018.

[123] Y. Nakayama *et al.*, "Low-latency routing for fronthaul network: A Monte Carlo machine learning approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

[124] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the Internet," in *Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, Karlsruhe, Germany, Aug. 2003, pp. 3–10.

[125] B.-N. Park, W. Lee, and C. Lee, "QoS-aware Internet access schemes for wireless mobile ad hoc networks," *Comput. Commun.*, vol. 30, no. 2, pp. 369–384, Jan. 2007.

[126] A. Chakraborty, L. E. Ortiz, and S. R. Das, "Network-side positioning of cellular-band devices with minimal effort," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 2767–2775.

[127] J. Zhan and M. Gastpar, "Functional forwarding of channel state information," *IEEE Trans. Inf. Theory*, vol. 60, no. 2, pp. 1008–1018, Feb. 2014.

[128] X. Chen, F. Mériaux, and S. Valentin, "Predicting a user's next cell with supervised learning based on channel states," in *Proc. IEEE 14th Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Darmstadt, Germany, Jun. 2013, pp. 36–40.

[129] S. Samulevicius, T. B. Pedersen, and T. B. Sorensen, "MOST: Mobile broadband network optimization using planned spatio-temporal events," in *Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring)*, Glasgow, U.K., May 2015, pp. 1–5.

[130] P. Sandhir and K. Mitchell, "A neural network demand prediction scheme for resource allocation in cellular wireless systems," in *Proc. IEEE Region 5 Conf.*, Kansas City, MO, USA, Apr. 2008, pp. 1–6.

[131] A. Adeel, H. Larijani, A. Javed, and A. Ahmadinia, "Critical analysis of learning algorithms in random neural network based cognitive engine for LTE systems," in *Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring)*, Glasgow, U.K., May 2015, pp. 1–5.

[132] P. Fazio, F. De Rango, and I. Selvaggi, "A novel passive bandwidth reservation algorithm based on neural networks path prediction in wireless environments," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst. (SPECTS)*, Ottawa, ON, Canada, Jul. 2010, pp. 38–43.

[133] M. Bkassiny, Y. Li, and S. K. Jayaweera, "A survey on machine-learning techniques in cognitive radios," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1136–1159, 3rd Quart., 2013.

[134] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive offloading for pervasive computing," *IEEE Pervasive Comput.*, vol. 3, no. 3, pp. 66–73, Jul.–Sep. 2004.

[135] B. Li and K. Nahrstedt, "A control-based middleware framework for quality-of-service adaptations," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 9, pp. 1632–1650, Sep. 1999.

[136] E. Altman, *Constrained Markov Decision Processes*. Boca Raton, FL, USA: CRC Press, 1999.

[137] A. Aral and I. Brandic, "Quality of service channelling for latency sensitive edge applications," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Honolulu, HI, USA, Jun. 2017, pp. 166–173.

[138] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Mach. Learn.*, vol. 29, nos. 2–3, pp. 131–163, Nov. 1997.

[139] D. G. Kleinbaum and M. Klein, *Logistic Regression: A Self-Learning Text*. New York, NY, USA: Springer-Verlag, 2010.

[140] T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Singapore, Dec. 2017, pp. 1–6.

[141] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-means clustering algorithm," *J. Roy. Stat. Soc. C (Appl. Stat.)*, vol. 28, no. 1, pp. 100–108, Jan. 1979.

[142] H. Du, S. Zeng, T. Dou, W. Fang, Y. Wang, and C. Zhang, "FASTBEE: A fast and self-adaptive clustering algorithm towards to edge computing," in *Proc. 5th IEEE Int. Conf. Cyber Security Cloud Comput. (CSCloud) 4th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Shanghai, China, Jun. 2018, pp. 128–133.

[143] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.

[144] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*. Boston, MA, USA: Springer, 2011, pp. 760–766.

[145] J. W. Weibull, *Evolutionary Game Theory*. Cambridge, MA, USA: MIT Press, 1997.

[146] H. Wang, R. Li, L. Fan, and H. Zhang, "Joint computation offloading and data caching with delay optimization in mobile-edge computing systems," in *Proc. 9th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Nanjing, China, Oct. 2017, pp. 1–6.

[147] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[148] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*. Amsterdam, The Netherlands: Springer, 1987.

[149] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 5, pp. 361–373, Sep. 2017.

[150] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, May 1992.

[151] T. Yang, Y. Hu, M. C. Gursoy, A. Schmeink, and R. Mathar, "Deep reinforcement learning based resource allocation in low latency edge computing networks," in *Proc. 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Lisbon, Portugal, Aug. 2018, pp. 1–5.

[152] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, Mar. 2016, pp. 2094–2100.

[153] T. Q. Dinh, Q. D. La, T. Q. S. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6353–6367, Dec. 2018.

[154] R. Dearden, N. Friedman, and S. Russell, "Bayesian Q-learning," in *Proc. 15th AAAI Conf. Artif. Intell.*, Madison, WI, USA, Jul. 1998, pp. 1–8.

[155] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190–10203, Nov. 2018.

[156] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nat. (Int. J. Sci.)*, vol. 518, pp. 529–533, Feb. 2015.

[157] Q. Gao, L. Gao, T. Xue, X. Zhu, X. Zhao, and R. Cao, "Multi-view learning based edge storage management strategy," in *Proc. 6th Int. Conf. Adv. Cloud and Big Data (CBD)*, Lanzhou, China, Aug. 2018, pp. 366–371.

[158] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of Machine Learning*. Boston, MA, USA: Springer, 2011, pp. 36–39.

[159] R. H. Riffenburgh, "Linear discriminant analysis," Ph.D. dissertation, Dept. Stat., Virginia Polytech. Inst. State Univ., Blacksburg, VA, USA, 1957.

[160] S. Wu *et al.*, "An efficient offloading algorithm based on support vector machine for mobile edge computing in vehicular networks," in *Proc. 10th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Hangzhou, China, Oct. 2018, pp. 1–6.

[161] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, Jul./Aug. 1998.

[162] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, MA, USA: MIT Press, 2004.

[163] Y. Yu, T. Hur, J. Jung, and I. G. Jang, "Deep learning for determining a near-optimal topological design without any iteration," *Struct. Multidiscipl. Optim. J.*, vol. 59, no. 3, pp. 787–799, Mar. 2019.

[164] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. D. Jesús, *Neural Network Design*. Boston, MA, USA: PWS, 1996.

[165] Z. Zhiwei, J. Li, and L. Xuebo, "Multi-parameter NCS scheduling based on fuzzy neural network," in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*, Xi'an, China, Aug. 2018, pp. 192–197.

[166] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. Int. Conf. Comput. Stat. (COMPSTAT)*, Sep. 2010, pp. 177–186.

[167] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Red Hook, NY, USA: Curran Assoc., 2010, pp. 2595–2603.

[168] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[169] H. Yi, S. Shiyu, D. Xiusheng, and C. Zhigang, "A study on deep neural networks framework," in *Proc. IEEE Adv. Inf. Manag. Commun. Electron. Autom. Control Conf. (IMCEC)*, Oct. 2016, pp. 1519–1522.

[170] *TensorFlow*. Accessed: May 5, 2019. [Online]. Available: https://www.tensorflow.org/

[171] *Google AI*. Accessed: May 5, 2019. [Online]. Available: https://ai.google/

[172] *DeepMind*. Accessed: May 5, 2019. [Online]. Available: https://deepmind.com/

[173] I. C. Dolcetta and H. Ishii, "Approximate solutions of the Bellman equation of deterministic control theory," *Appl. Math. Optim.*, vol. 11, no. 1, pp. 161–181, Feb. 1984.

[174] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2010, pp. 1–8.

[175] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chenometr. Intell. Lab. Syst.*, vol. 2, no. 1, pp. 37–52, Aug. 1987.

[176] Y. Ding and J. Liu, "Real-time false data injection attack detection in energy Internet using online robust principal component analysis," in *Proc. IEEE Conf. Energy Internet Energy Syst. Integr. (EI2)*, Nov. 2017, pp. 1–6.

[177] A. Fabris, M. A. Nicolaou, I. Kotsia, and S. Zafeiriou, "Dynamic probabilistic linear discriminant analysis for video classification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 2781–2785.

[178] Z. Yang, Y. Xie, and Z. Wang. (May 2019). *A Theoretical Analysis of Deep Q-Learning*. Accessed: Aug. 7, 2019. [Online]. Available: https://arxiv.org/abs/1901.00137

[179] W. Yu, R. Wang, R. Li, J. Gao, and X. Hu, "Historical best Q-networks for deep reinforcement learning," in *Proc. IEEE 30th Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2018, pp. 6–11.

[180] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Assoc., 2018, pp. 8778–8788.

[181] F. D. Vita, D. Bruneo, A. Puliafito, G. Nardini, A. Virdis, and G. Stea, "A deep reinforcement learning approach for data migration in multi-access edge computing," in *Proc. ITU Kaleidoscope Mach. Learn. 5G Future (ITU K)*, Nov. 2018, pp. 1–8.

[182] R. Parihar, A. Jain, and U. Singh, "Support vector machine through detecting packet dropping misbehaving nodes in MANET," in *Proc. Int. Conf. Electron. Commun. Aerosp. Technol. (ICECA)*, Apr. 2017, pp. 483–488.

[183] B. Scholkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, U.K.: MIT Press, 2001.

[184] E. Osuna and F. Girosi, "Reducing the run-time complexity of support vector machines," in *Proc. IAPR Int. Conf. Pattern Recognit. (ICPR)*, Aug. 1998, pp. 1–10.

[185] Y. Tang. (Feb. 2015). *Deep Learning Using Linear Support Vector Machines*. Accessed: May 16, 2019. [Online]. Available: https://arxiv.org/abs/1306.0239

[186] A. Abdiansah and R. Wardoyo, "Time complexity analysis of support vector machines (SVM) in LibSVM," *Int. J. Comput. Appl.*, vol. 128, no. 3, pp. 28–34, Oct. 2015.

[187] H. Song, Z. Ding, C. Guo, Z. Li, and H. Xia, "Research on combination kernel function of support vector machine," in *Proc. Int. Conf. Comput. Sci. Softw. Eng.*, Dec. 2008, pp. 838–841.

[188] R. G. J. Wijnhoven and P. H. N. de With, "Fast training of object detection using stochastic gradient descent," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 424–427.

[189] P. M. Lee, *Bayesian Statistics*. London, U.K.: Wiley, 1997.

[190] R. W. Strachan and H. K. van Dijk, "Bayesian model selection with an uninformative prior," *Oxford Bull. Econ. Stat.*, vol. 65, no. 1, pp. 863–876, Dec. 2003.

[191] A. Gelman, "Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper)," *Bayesian Anal.*, vol. 1, no. 3, pp. 515–534, Sep. 2006.

[192] N. L. Hjort, C. Homes, P. Müller, and S. G. Walker, *Bayesian Nonparametrics*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[193] S. Brooks, A. Gelman, G. L. Jones, and X. L. Meng, *Handbook of Markov Chain Monte Carlo*. Boca Raton, FL, USA: CRC Press, 2011.

[194] P. G. Constantine, M. S. Eldred, and E. T. Philipps, "Sparse pseudospectral approximation method," *Comput. Methods Appl. Mech. Eng.*, vols. 229–232, pp. 1–12, Jul. 2012.

[195] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by Gibbs sampling," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguist.*, Jun. 2005, pp. 363–370.

[196] H. Haario, E. Saksman, and J. Tamminen, "An adaptive metropolis algorithm," *Bernoulli*, vol. 7, no. 2, pp. 223–242, Apr. 2001.

[197] T. Konishi, T. Kubo, K. Watanabe, and K. Ikeda, "Variational Bayesian inference algorithms for infinite relational model of network data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 9, pp. 2176–2181, Sep. 2015.

[198] A. M. Alaa and M. van der Schaar, "Bayesian nonparametric causal inference: Information rates and learning algorithms," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 5, pp. 1031–1046, Oct. 2018.

[199] W. Meng, K.-K. R. Choo, S. Furnell, A. V. Vasilakos, and C. W. Probst, "Towards Bayesian-based trust management for insider attacks in healthcare software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 2, pp. 761–773, Jun. 2018.

[200] Z. Pawlak, "Rough sets, decision algorithms and Bayes' theorem," *Eur. J. Oper. Res.*, vol. 136, no. 1, pp. 181–189, Jan. 2002.

[201] B. Guermah, T. Sadiki, and H. E. Ghazi, "Fuzzy logic approach for GNSS signal classification using RHCP and LHCP antennas," in *Proc. IEEE 8th Annu. Ubiquitous Comput. Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2017, pp. 203–208.

[202] J. Yen and R. Langari, *Fuzzy Logic: Intelligence, Control and Information*. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[203] J. Kwisthout, "Most probable explanations in Bayesian networks: Complexity and tractability," *Int. J. Approx. Reason.*, vol. 52, no. 9, pp. 1452–1469, Dec. 2011.

[204] C. Bielza and P. Larrañaga, "Discrete Bayesian network classifiers: A survey," *ACM Comput. Surveys*, vol. 47, no. 1, pp. 1–43, Jul. 2014.

[205] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31–37, Dec. 2017.

[206] V. Torra, "Hesitant fuzzy sets," *Int. J. Intell. Syst.*, vol. 25, no. 6, pp. 529–539, Mar. 2010.

[207] B. Li, T. Chen, X. Wang, and G. B. Giannakis, "Secure edge computing in IoT via online learning," in *Proc. 52nd Asilomar Conf. Signals Syst. Comput.*, Oct. 2018, pp. 2149–2153.

[208] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Proc. Eur. Conf. Mach. Learn.*, Oct. 2005, pp. 437–448.

[209] A. Carrega, M. Repetto, P. Gouvas, and A. Zafeiropoulos, "A middleware for mobile edge computing," *IEEE Cloud Comput.*, vol. 4, no. 4, pp. 26–37, Jul. 2017.

[210] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy K-nearest neighbor algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 4, pp. 580–585, Jul./Aug. 1985.

[211] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[212] Y. Liu *et al.*, "Towards a smart campus: Innovative applications with WiCloud platform based on mobile edge computing," in *Proc. 12th Int. Conf. Comput. Sci. Educ. (ICCSE)*, Aug. 2017, pp. 133–138.

[213] C. Tesgera, M. Klein, and A. Juan-Verdejo, "A cloudlet-based approach to tackle network challenges in mobile cloud applications," in *Proc. 14th Int. Conf. Adv. ICT Emerg. Regions (ICTer)*, Dec. 2014, p. 253.

[214] Z. Bai and J. Qin, "Design of mobile server framework in MANET," in *Proc. 5th Int. Conf. Wireless Commun. Netw. Mobile Comput.*, Sep. 2009, pp. 1–4.

[215] U. Ali and T. Ahmed, "Mobile chatting server for GPRS networks," in *Proc. Int. Conf. Emerg. Technol.*, Nov. 2007, pp. 52–57.

[216] M. Aziz, U. Akram, K. Rashed, and M. Jarke, "QoS framework for mobile-to-mobile multimedia streaming applications," in *Proc. Sci. Inf. Conf. (SAI)*, Jul. 2015, pp. 1011–1017.

[217] M. Narang *et al.*, "UAV-assisted edge infrastructure for challenged networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2017, pp. 60–65.

[218] K. H. Low, "An initial parametric study of weight and energy thresholds for falling unmanned aerial vehicles (UAVs)," in *Proc. Workshop Res. Educ. Develop. Unmanned Aerial Syst. (RED-UAS)*, Oct. 2017, pp. 240–245.

[219] G. Laporte and S. Martello, "The selective travelling salesman problem," *Discr. Appl. Math.*, vol. 26, nos. 2–3, pp. 193–207, Mar. 1990.

[220] L. Valerio, A. Passarella, and M. Conti, "Accuracy vs. traffic trade-off of learning IoT data patterns at the edge with hypothesis transfer learning," in *Proc. IEEE 2nd Int. Forum Res. Technol. Soc. Ind. Leveraging Better Tomorrow (RTSI)*, Sep. 2016, pp. 1–6.

[221] K. Xing, C. Hu, J. Yu, X. Cheng, and F. Zhang, "Mutual privacy preserving k -means clustering in social participatory sensing," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 2066–2076, Aug. 2017.

[222] R. Radhakrishnan, W. W. Edmonson, F. Afghah, R. M. Rodriguez-Osorio, F. Pinto, and S. C. Burleigh, "Survey of inter-satellite communication for small satellite systems: Physical layer to network layer view," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2442–2473, 4th Quart., 2016.

[223] T. Li, H. Zhou, H. Luo, Q. Xu, S. Hua, and B. Feng, "Service function chain in small satellite-based software defined satellite networks," *China Commun.*, vol. 15, no. 3, pp. 157–167, Mar. 2018.

[224] J. A. Ruiz-De-Azúa, A. Camps, and A. C. Augé, "Benefits of using mobile ad-hoc network protocols in federated satellite systems for polar satellite missions," *IEEE Access*, vol. 6, pp. 56356–56367, 2018.

[225] S. Li and F. Tang, "Load-balanced cooperative transmission in MEO-LEO satellite network," in *Proc. IEEE 32nd Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, May 2018, pp. 564–571.

[226] H. Gedawy, K. Habak, K. A. Harras, and M. Hamdi, "Awakening the cloud within: Energy-aware task scheduling on edge IoT devices," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2018, pp. 191–196.

[227] Y. Xu, K. Li, T. T. Khac, and M. Qiu, "A multiple priority queueing genetic algorithm for task scheduling on heterogeneous computing systems," in *Proc. IEEE 14th Int. Conf. High Perform. Comput. Commun. 9th Int. Conf. Embedded Softw. Syst.*, Jun. 2012, pp. 639–646.

[228] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo, "Self-organized task allocation to sequentially interdependent tasks in swarm robotics," *Auton. Agents Multi Agent Syst.*, vol. 28, no. 1, pp. 101–125, Jan. 2014.

Tiago Koketsu Rodrigues (M'15) received the bachelor's degree in computer science from the Federal University of Piaui, Brazil, in 2014, and the M.Sc. degree from Tohoku University, Japan, in 2017, where he is currently pursuing the Ph.D. degree. He was a Researcher with Bucknell University, USA, in 2013, and Tohoku University in 2014. He has previously been awarded a scholarship by the Coordination for the Improvement of Higher Education Personnel from Brazil to study for one year in Bucknell University in 2013 and a scholarship from the Japanese Ministry of Education, Culture, Sports, Science, and Technology to pursue his master's degree in Japan. He is currently a Scholar under the Japan Society for the Promotion of Science. He was a recipient of the Best Presentation Award in the A3 Foresight Program 2016 Annual Workshop, the IEEE VTS Japan 2016 Young Researcher's Encouragement Award, and the 2017 Tohoku University Graduate School of Information Sciences Dean Award. Since 2017, he has been the System Administrator of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, overviewing the review process of all submissions and the submission system as a whole.

Katsuya Suto (M'16) received the B.Sc. degree in computer engineering from Iwate University, Morioka, Japan, in 2011, and the M.Sc. and Ph.D. degrees in information science from Tohoku University, Sendai, Japan, in 2013 and 2016, respectively. He was a Post-Doctoral Fellow with the Broadband Communications Research Laboratory, University of Waterloo, ON, Canada, from 2016 to 2018. He is currently an Assistant Professor with the Department of Computer and Network Engineering, University of Electro-Communications, Tokyo, Japan. His research interests include software defined networking, mobile edge computing, and Internet of Things. He was a recipient of the Best Paper Award at the IEEE VTC 2013-Spring, IEEE/CIC ICCC 2015, and IEEE ICC 2016.

Hiroki Nishiyama (SM'13) received the M.S. and Ph.D. degrees in information science from Tohoku University, Japan, in 2007 and 2008, respectively, where he is a Professor with the Graduate School of Engineering. He has published over 190 peer-reviewed papers, including many high quality publications in prestigious IEEE journals and conferences. His research interests cover a wide range of areas, including satellite communications, unmanned aircraft system networks, wireless and mobile networks, ad-hoc and sensor networks, green networking, and network security. He was a recipient of the Best Paper Awards from many international conferences including IEEE's flagship events, such as the IEEE Global Communications Conference in 2014 (GLOBECOM'14), GLOBECOM'13, and GLOBECOM'10, the IEEE International Conference on Communications in 2018 (ICC'18), ICC'17, and ICC'16, the IEEE Wireless Communications and Networking Conference in 2014 (WCNC'14) and WCNC'12, the Prizes for Science and Technology from the Minister of Education, Culture, Sports, Science, and Technology, Japan, in 2018, the 2017 FUNAI Foundation's Academic Award for Information Technology, the 29th Advanced Technology Award for Creativity in 2015, the IEEE Communications Society Asia–Pacific Board Outstanding Young Researcher Award 2013, and the IEICE Communications Society Academic Encouragement Award 2011. He currently serves as the Secretary for IEEE ComSoc Sendai Chapter. One of his outstanding achievements is Relay-by-Smartphone, which makes it possible to share information among many people by device-to-device direct communication. He is a Senior Member of the Institute of Electronics, Information and Communication Engineers.

Jiajia Liu (SM'15) has been a Full Professor with the School of Cyber Engineering, Xidian University since 2013, and has been selected into the prestigious "Huashan Scholars" program by Xidian University since 2015. He has published over 90 peer-reviewed papers in many high quality publications, including prestigious IEEE journals and conferences. His research interests cover a wide range of areas, including load balancing, wireless and mobile ad-hoc networks, FiberWireless networks, Internet of Things, network security, LTE-A and 5G, SDN, and NFV. He was a recipient of the Best Paper Awards from many international conferences including IEEE flagship events, such as IEEE WCNC in 2012 and 2014. He has been actively joining the society activities, such as serving as an Associate Editor for the IEEE TRANSACTIONS ON COMPUTERS and the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, an Editor for the IEEE NETWORK and the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the Guest Editor of top ranking international journals, such as the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING and the IEEE INTERNET OF THINGS JOURNAL, and serving as the technical program committees of numerous international conferences. He is a Distinguished Lecturer of IEEE Communications Society.

Nei Kato (F'13) is a Full Professor (Deputy Dean) with the Graduate School of Information Sciences and the Director of the Research Organization of Electrical Communication, Tohoku University, Japan. He has published over 400 papers in prestigious peer-reviewed journals and conferences. He has been engaged in research on computer networking, wireless mobile communications, satellite communications, ad-hoc and sensor and mesh networks, smart grid, AI, IoT, big data, and pattern recognition. He was a recipient of several awards, including the Minoru Ishida Foundation Research Encouragement Prize in 2003, the Distinguished Contributions to Satellite Communications Award from the IEEE Communications Society, Satellite and Space Communications Technical Committee in 2005, the FUNAI Information Science Award in 2007, the TELCOM System Technology Award from Foundation for Electrical Communications Diffusion in 2008, the IEICE Network System Research Award in 2009, the IEICE Satellite Communications Research Award in 2011, the KDDI Foundation Excellent Research Award in 2012, the IEICE Communications Society Distinguished Service Award in 2012, the IEICE Communications Society Best Paper Award in 2012, the Distinguished Contributions to Disaster-Resilient Networks R&D Award from Ministry of Internal Affairs and Communications, Japan, in 2014, the Outstanding Service and Leadership Recognition Award 2016 from IEEE Communications Society Ad Hoc and Sensor Networks Technical Committee, the Radio Achievements Award from Ministry of Internal Affairs and Communications, Japan, in 2016, the IEEE Communications Society Asia–Pacific Outstanding Paper Award in 2017, the Prize for Science and Technology from the Minister of Education, Culture, Sports, Science, and Technology, Japan, in 2018, the Award from Tohoku Bureau of Telecommunications, Ministry of Internal Affairs and Communications, Japan, in 2018, and the Best Paper Awards from IEEE ICC/GLOBECOM/WCNC/VTC. He has been the Vice President (Member and Global Activities) of the IEEE Communications Society since 2018, the Editor-in-Chief of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY since 2017, and the Chair of IEEE Communications Society Sendai Chapter. He served as the Editor-in-Chief for the *IEEE Network Magazine* from 2015 to 2017, a Member-at-Large on the Board of Governors, IEEE Communications Society from 2014 to 2016, the Vice Chair of the Fellow Committee of IEEE Computer Society in 2016, and a member of IEEE Communications Society Award Committee from 2015 to 2017. He has also served as the Chair of Satellite and Space Communications Technical Committee from 2010 to 2012 and Ad Hoc and Sensor Networks Technical Committee from 2014 to 2015 of IEEE Communications Society. He is a Distinguished Lecturer of the IEEE Communications Society and Vehicular Technology Society. He is a fellow of the Engineering Academy of Japan and IEICE.