IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 May 2022, accepted 5 June 2022, date of publication 1 July 2022, date of current version 11 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3187701

A Review of Intelligent Computation Offloading in Multiaccess Edge Computing

HENGLI JIN, (Student Member, IEEE), MARK A. GREGORY, (Senior Member, IEEE), AND SHUO LI, (Member, IEEE)
School of Engineering, RMIT University, Melbourne, VIC 3000, Australia
Corresponding author: Mark A. Gregory (mark.gregory@rmit.edu.au)

ABSTRACT Multi-Access Edge Computing (MEC) is a standardized architecture that enables cloud computing capabilities at the edge of heterogeneous networks. The concept is to reduce network congestion by running applications and network services closer to end-users. MEC is designed to be implemented at key locations on the network edge, including co-location with cellular base stations. MEC aims to facilitate computation intensive and delay sensitive applications, such as vehicular networks, face recognition, augmented reality and virtual reality. The service requirements for MEC are stochastic and time varying. Coupled with advances in artificial intelligence, a vast number of computation offloading approaches have been developed based on intelligent algorithms. This article provides a comprehensive review of intelligent computation offloading with critical issues, metrics and future directions.

INDEX TERMS Multi-access edge computing, computation offloading, artificial intelligence, computer networking.

I. INTRODUCTION

With the advent of the 5G technologies and the proliferation of computation and storage devices [1], new applications and scenarios have emerged, such as smart real-time navigation system applications or the ubiquitous Internet of Things (IoT) devices that have in-built sensors, which typically have high energy consumption, and generate data that is processed upstream. There is a need to reduce the network traffic from the edge to the cloud by introducing massively distributed computing capability with strict latency requirements [2]. To this end, Multi-Access Edge Computing (MEC) was introduced as a new paradigm that complements and extends cloud computing to the network edge. MEC provides computing and storage with the aim to carry out resource-intensive tasks offloaded by the end-user devices and to carry out IoT data collection, processing and aggregation [3], thus alleviating the transit network workload and enhancing the end-user quality of experience [4], [5].

MEC was defined initially as 'mobile-edge computing' in 2014 by the European Telecommunication Standards Institute (ETSI) and encompassed non-mobile capabilities in 2017 when 'mobile' became 'multi-access' [6] to reflect the additional use-cases that MEC would address including computing and storage support for end-user devices over fixed access technologies, e.g., fibre, satellite, wireless, fibre-wireless, and light-fibre. MEC development has embraced Software Defined Networking (SDN) and virtualization [7]. SDN and Network Function Virtualization (NFV) are a logical addition to MEC nodes as they provide intelligent network integration and resource optimization capability [8]. The MEC nodes utilise virtualization to host third party Virtual Machines (VM) and containers forming a multi-tenant ecosystem on the network edge [9], [10]. As MEC nodes are located on the network edge, there is a need to facilitate service mobility, and VM handoff techniques [11] and container-based handoff techniques [12] can be found in the literature. The original definition of MEC focused on providing support for mobile cellular networks and the need to provide computation offloading support to mobile cellular Base Stations (BS) and User Equipment (UE). As other scenarios were explored the research community introduced fog computing and mist computing. To some extent, MEC now incorporates fog computing and other scenarios.

The associate editor coordinating the review of this manuscript and approving it for publication was Ali Kashif Bashir.

Computation offloading is the transfer of computation intensive tasks from the UE to an edge server or to a cloud server to achieve an optimal balance between the execution time and energy power consumption [13]. Fig. 1 depicts task offloading in an MEC environment. The workflow starts with the UE execution request. The applications that support and benefit from computation offloading are prioritized using the relevant metrics, e.g., latency, resource availability and energy consumption. For the low priority applications, task computation is likely to occur on the UE. The offloading scheduling in the edge server is a sophisticated process to cope with the constant dynamic multi-access heterogeneous networks. The computation offloading approach is usually a joint optimization solution that considers the offloading decision, resource allocation, mobility management, content caching, security, and privacy. It can be treated as a multi-dimension and multi-objective optimization problem and is known to be non-deterministic polynomial time (NP-hard) [14], [15].

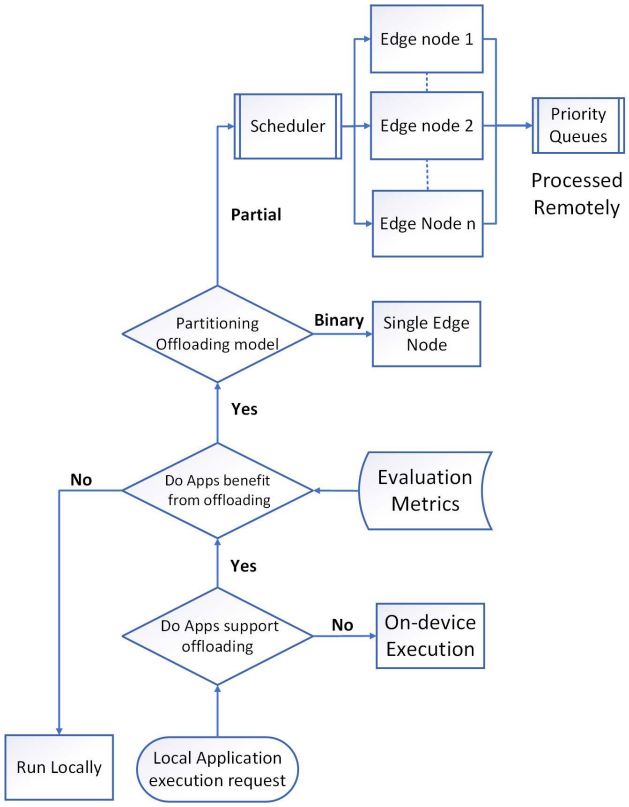

FIGURE 1. Computation offloading process.

Traditional optimization approaches, such as convex optimization, usually require prior knowledge about mobile user patterns and network parameters. The traditional optimization approaches are more suitable for static networks or slowly varying environments [16]. By contrast, artificial intelligence (AI) approaches provide a means to tackle complex optimization problems [17], [18]. In a typical MEC deployment intelligent computation offloading can predict, for different computation tasks, demand in terms of type, size, and computing resource requirements. Intelligent computation offloading adapts to dynamic network environments to deal with the changing computation tasks and network resource demands, while ensuring robustness and computational tractability [19], [20].

The AI algorithms that are applied to MEC optimization are generally divided into three categories: Machine Learning (ML), Evolutionary Algorithms (EA), and Swarm Intelligent Algorithms (SIA). There are other practical algorithms, such as Fuzzy [21], Intelligent Reflecting Surface [22] and Game-Theoretic Learning [23], that have been applied to MEC computation offloading optimization and the results were encouraging. Nevertheless, the three categories are more prevalent in recent research [24], [25]. In particular, the Genetic Algorithm (GA) [26]–[29] and Ant colony algorithms [30], [31] as well as Particle Swarm Optimization [32] have been applied widely for job scheduling optimization. GAs have various crossover and mutation operators that can deal with the discrete and continuous optimization problems. The ant colony algorithm (ACO) can be assigned to VMs to optimize the scheduling process [33]. Deep Reinforcement Learning (DRL) in ML has been applied to resource allocation, task offloading and combinational problems [33]–[35], DRL is adaptive and efficiently learns from experiences or data sets.

The MEC paradigm has attracted the attention of researchers in both academia and industry. MEC related review and survey papers have been published that cover characteristics, framework, enabling techniques, use cases, implementation, standardization, and connections with IoT and 5G [36], [37]. For an overview of MEC, the author in [38] presents the related concepts, a brief background on edge computing and showcases computation offloading application scenarios. The survey [9] focuses on the enabling techniques in MEC and includes a description of how MEC with VM, NFV, and SDN provide flexible control and multitenancy support. The author in [1] described three MEC case studies: mobile edge orchestration, collaborative video caching and processing two-layer interference cancellation. Meanwhile, the author in [13] presents an overview on task offloading and classified existing works as greedy heuristics, integer programming, machine learning branch and bound and convex optimization.

The article [24] reviewed the applications of communication intelligence techniques to four critical issues in cloud computing and edge computing: job scheduling, resource allocation, task offloading, and joint issues. The authors of [20] surveyed applications of computational intelligence techniques along with five different problems related to wireless sensor and actuator networks: actuation, communication, sink mobility, topology control and localization. The convergence of AI and edge computing was discussed in [39] with a focus on the AI approaches for edge intelligence. The authors propose a broader perspective that distinguishes edge intelligence into AI for the edge and AI on the edge by presenting

state of the art work and explaining how to carry out the training and inference of AI models on the edge. Similarly, the survey [40] introduced and discussed scenarios and fundamental enabling techniques for edge intelligence and reviews the deployment issues of DL for edge computing, spanning networks, communication, and computation. As one of the mainstream AI techniques, ML-based approaches have been widely used in edge computing. The review paper [16] briefly introduced, compared and summarized the basic characteristics, prominent advantages, limitations, and typical applications of ML-based approaches used in MEC offloading. The surveys [33], [41] focus on the ML-based computation offloading mechanisms in the MEC environment, [41] points out limitations for some algorithms, [33] classified ML algorithms into three main categories: supervised learning, unsupervised learning, and reinforcement learning, and proposed a taxonomy of offloading ML-based mechanisms. Potential business opportunities related to MEC were discussed in [42] from the perspectives of application developers, service providers, and network equipment vendors.

Recent surveys on the edge computing paradigm [38]–[41] highlight that MEC is expected to adopt intelligent scheduling with self-optimization and self-adaption, which is not only applied in protocols or algorithms design. The joint optimization to construct a systematic framework can be realized with the aid of AI [31]. The ML breakthroughs are pushing AI closer to the edge computing environment. To this end, this paper provides a review on intelligent approaches in MEC computation offloading, identifies the use case and metrics. The challenges that remain are discussed. The contributions of this review paper are:

- An outline of the MEC architecture from the UEs and edge perspectives. In particular, metrics and related critical issues about computation offloading.
- To provide an overview of intelligent offloading, this research identifies the concepts and uses cases in the three categories: ML, EA, and SIA.
- Discusses the remaining challenges for intelligent optimisation in MEC.

The rest of this paper is as follows. Section II describes the computation offloading process in an MEC network. Section III investigates AI based computation offloading approaches. Section IV discusses future research challenges. The paper is concluded in Section V.

II. COMPUTATION OFFLOADING IN THE MEC ARCHITECTURE

As shown in the typical MEC architecture (Fig.2), the end device consumes and produces data [35]. At the edge, MEC consists of hardware, software and networking resources. Virtualization layers are employed to manage the resources based on network and task functions. The hardware resources provide the computing, storage and control functionality. Virtualization is ued to provide computing, storage, caching, virtual exchange, networking and corresponding management functions. The computation solutions are generally defined by the MEC system model assumptions and reference architecture: Single MEC server colocated with a mobile cellular network BS, Multiple MEC servers colocated with a BS and one or more MEC servers colocated with one or more BS [13].

A. MEC METRICS

An optimal edge computing load distribution strategy can be developed using multiple allocation strategies for the heterogeneous network and generally utilizes four metrics.

1) LATENCY

Latency is a metric that is important for time sensitive applications and services. It is not only determined by application or service processing time at the server but it is also affected by the transmission time to and from the UE.

2) BANDWIDTH

Bandwidth requirements and availability make it a key metric as it can either reduce or increase transmission delay. Transmission technologies, e.g., wireless and optical, can have a fundamental impact on bandwidth availability and transmission delay. As mobility is an important requirement for end users, particularly when using mobile cellular devices, it is helpful to position MEC servers at the network edge to reduce the need for UE to connect directly to the centralized cloud computing resources.

3) POWER CONSUMPTION

For terminal devices, migrating loads to the network edge can save energy. Therefore, there is a trade-off between local computing and transmission power consumption. It is necessary to determine whether the energy consumption characteristics of the load are computationally intensive and how much computing power is required. Besides the transmission energy consumption and network signal strength, the data size and available bandwidth affect transmission energy consumption.

4) COST

For the service provider, edge computing can guarantee reduced latency and lower energy consumption, thus increasing throughput, improving user experience and ultimately reducing costs.

In general, the correlation between the metrics should be considered for computation offloading and resource allocation. Optimization metrics should be weighted and prioritized for different workloads so that the system can identify the allocation strategy.

B. COMPUTATIONAL TASK MODEL

From a computer programming perspective, the essence of a program is a list of tasks that the computer completes step-by-step to achieve a specific outcome. In other words, a task is a logical unit of a program that contains a set

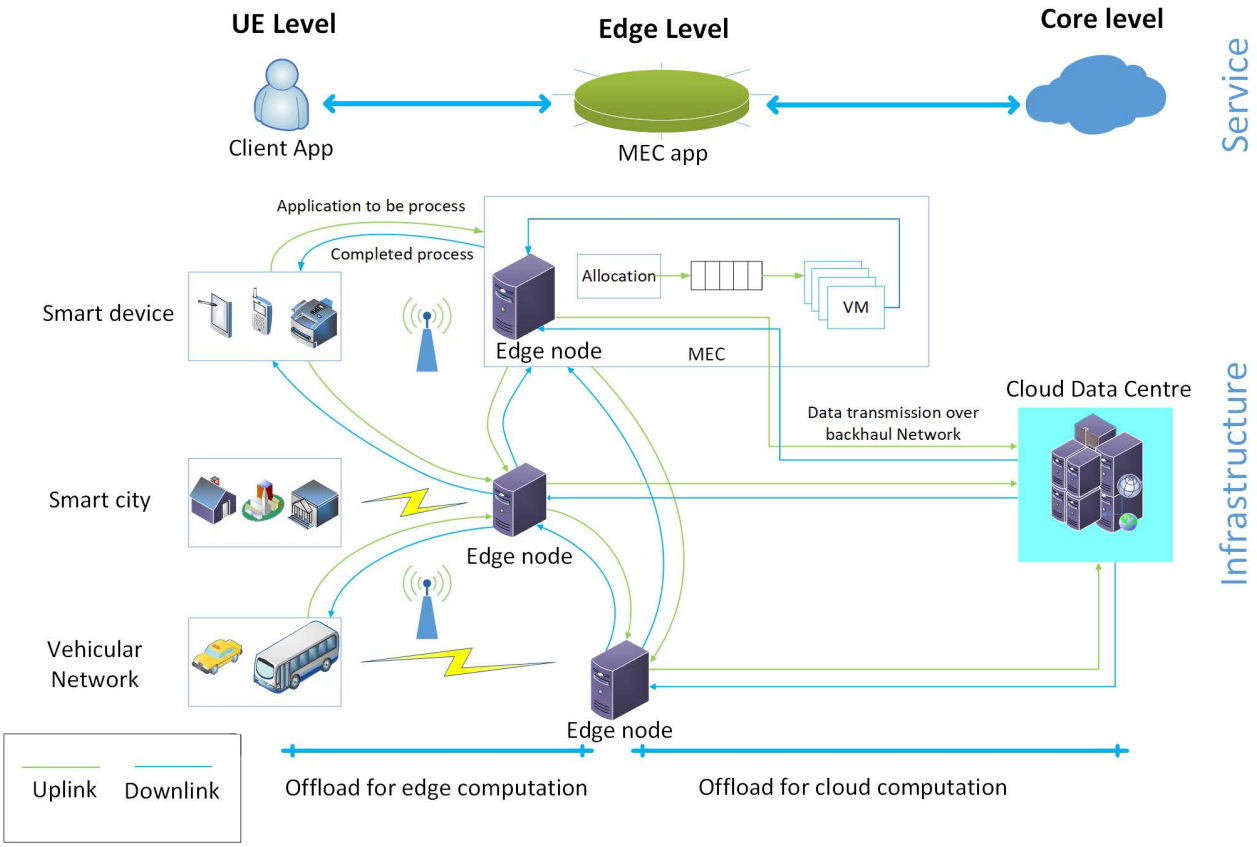

FIGURE 2. MEC architecture [16].

of coupled instructions. Depending on how an application is developed, a task can be a complete program or a collection of one or more functions within a program. Since a user program can cooperate with the services of other utilities, a utility program can also be considered a separate task (or subtask). The MEC computational offload models includes both binary and partial offloads. Among them, the binary offload computation model is suitable for highly integrated computation tasks with indivisible data, which require the user to offload the task either locally or to a single MEC server. The key parameters for this type of computing task can be represented by a three-field notation A(L,T_d,X) with the amount of data to be computed L (number of bits), the computational intensity X (number of CPU clock cycles required per bit of data), and the computational latency requirements T_d (completion time in seconds) [43]. The parameters are related to the specific computational task and can be derived by profiling and modelling the computational task. The partitionable tasks can be mainly categorised as data-partitioned oriented, code-partitioned-oriented, and continuous-execution tasks [44]. For those tasks that can be executed in parallel, the offloading strategy only considers computation resource allocation. However, the process should include queuing and scheduling for those dependency required tasks.

C. LOCAL EXECUTION COMPUTATION OFFLOADING

MEC computation offloading strategies are determined by considering the decision strategy, resource allocation, and mobility management. From the user perspective, the primary function of computation offloading is to minimize the energy usage and speed up task execution time [45]. For binary or partial offloading, the decision process can be seen as an optimization problem that combines wireless channel selection and energy allocation while considering the network state variables for the UE concurrently, which involves the task queuing state, the cumulative UE energy consumption, the occupying wireless channel, and available wireless channel quality [46].

The author in [47] considers utilizing Time Division Multiple Access (TDMA) for an arbitrary time slot allocation. The transmission rate is calculated as:

$$r_k = B \log_2 \left(1 + \frac{p_k h_k^2}{N_0}\right) \tag{1}$$

B and N_0 are the bandwidth and the variance of complex white Gaussian channel noise, and p_k and h_k are UE k's transmission power and channel gain respectively.

Orthogonal Frequency Division Multiple Access (OFDMA) takes into account n sub-channels, which can be denoted by a set $N = \{1, 2, , N\}$ [48]. If h_k and N_0 are considered constant,

the function (1) is concave (convex). Otherwise,the random processes of white Gaussian channel noise and channel gain should be considered [49]. In practice, the interference model, e.g., Code Division Multiple Access (CDMA), should also be considered as one of the channel models. The transmission rates are calculated as [50]:

$$r_k = B\log_2\left(1+\frac{p_k h_k^2}{N_0+\sum_{i\neq k} p_i h_i^2}\right) \tag{2}$$

The $\sum_{i\neq k} p_i h_i^2$ denotes the interference from other UEs.

Applications with high computational demand that have low data transmission needs are more suitable for offloading [51]. Local computation energy consumption is mainly association with CPU power consumption, which is generally inversely proportional to the square of the CPU operating frequency. CPU performance is controlled by the clock cycle frequency (also known as clock speed). Mainstream mobile device CPU architectures employ advanced Dynamic Frequency and Voltage Scaling (DVFS) techniques, which permit the CPU cycle frequency (or voltage) to be increased or decreased, thereby increasing and decreasing power consumption, respectively. For local computation, the computation latency is proportional to the number of CPU clock cycles required for the computation process to complete and inversely proportional to the CPU's operating frequency [52]; therefore, the local computation latency can be reduced by increasing the CPU's operating frequency, however, energy consumption increases as a result. It is cost effective to compute locally, especially if the channel quality between the UE and the nearest computation offloading server is low [53]. Data transmission energy consumption is expensive at the UE. For applications that support partitioning, a partial offloading approach can reduce UE energy consumption when compared to a binary offloading approach [54].

The radio access technology used for the radio link between access point and UE can affect energy minimization strategies [45]. UE energy savings increase when OFDMA is used when compared with TDMA due to the higher granularity of radio resources [55]. Emerging techniques including wireless power transfer [56] and non-orthogonal multiple access (NOMA) should further enhance the performance of MEC [57], [58].

D. CLUSTERING ALGORITHMS

All tasks have latency sensitivities, thus tasks can be clustered using latency sensitivity as a factor. The clustering process aids with the computation offloading scheduling of the tasks. Clusters are prioritized and within each cluster, the tasks are ranked in priority order so that latency-sensitive tasks can be actioned first, thus effectively reducing the queuing latency of subsequent tasks. The role of clustering algorithms, like unsupervised learning algorithms, is to divide a data set into multiple subsets, or clusters. Clustering is the process of using an algorithm to divide samples in a dataset into disjoint subsets. The purpose of clustering algorithms is to classify a data set into clusters that satisfy two conditions: the tasks in the same cluster are similar based on pre-determined criteria, and the tasks in different clusters are different based on pre-determined criteria. A common approach for determining the similarity between points in clustering algorithms is to use the Euclidean and Mahalanobis distances between two points. The Euclidean distance in n dimensional feature space can be denoted as:

$$d_{x,y} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3}$$

Mahalanobis distance can be denoted as:

$$d_{x,y} = \sum_{\mathrm{i}=1}^{\mathrm{n}} |\mathrm{x_i} - \mathrm{y_i}| \tag{4}$$

Minkowski Distance is a generalization of Mahalanobis distance and can be expressed as:

$$d_{x,y} = \left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p} \tag{5}$$

It can be seen that for the Ming distance, when $p = 1$, it corresponds to the Mahalanobis distance, and when $p = 2$, it corresponds to Euclidean distance.

For the task clustering problem in the MEC system, we need to determine the metrics of task similarity before classifying tasks. Assume that there are N tasks numbered as $\{S_1, S_2, S_3, \ldots, S_N\}$. The total amount of computation consumed to complete each task is respectively $\{L_1, L_2, L_3, \ldots, L_N\}$. The sensitivity of all tasks to delay can be divided into M categories, numbered as $\{1, 2, 3, \ldots, M\}$. The delay sensitivity of N tasks can be expressed as $\{P_1, P_2, P_3, \ldots, P_N\}$, For $P_i \in \{1, 2, 3, \ldots, M\}, i = \{S_1, S_2, S_3, \ldots, S_N\}$ The classified cluster can define as $\{C_1, C_2, C_3, \ldots, C_N\}$. A minimal maximum standard deviation of the clusters, obtained after classification, is sought.

At a particular time, the task in the SDN controller is $\{S_1^*, S_2^*, \ldots, S_n^*\}$. The corresponding time delay sensitivity is $\{P_1', P_2', \ldots, P_n'\}$ respectively. After the normalization of the delay sensitivity, the delay sensitivity obtained is: $\{P_1^*, P_2^*, \ldots, P_n^*\}$.

$$P_i^* = \frac{P_i'}{\sum_{j=1}^{n} P_j'} \tag{6}$$

The calculations consumed to complete the task is $\{L_1', L_2', \ldots, L_n'\}$. The calculations consumed after the normalization operation is $\{L_1^*, L_2^*, \ldots, L_n^*\}$.

$$\mathrm{L_i^*} = \frac{\mathrm{L_i'}}{\sum_{\mathrm{j}=1}^{\mathrm{n}} \mathrm{L_j'}} \tag{7}$$

The Euclidean distance between S_i^* and S_j^* in the same cluster can define as

$$d_{i,j} = \sqrt{\left(P_i^* - P_j^*\right)^2 + \left(L_i^* - L_j^*\right)^2} \tag{8}$$

The standard deviation corresponding to tasks in the same cluster is defined as D, when the cluster C_i have the tasks $\{S_1^*, S_2^*, \ldots, S_n^*\}$. The corresponding standard deviation is:

$$D_i = \sqrt{\frac{\sum_{i=1}^{n'} \sum_{j=1, j\neq i}^{n'} \left(d_{i,j} - \bar{d}\right)^2}{C_{n'}^2}} \tag{9}$$

In the same cluster, the smaller the standard deviation of all tasks, the higher the similarity of the tasks. Therefore, when all tasks are divided into K clusters, the optimal result is given by:

$$\min \{\max D_i\}, i = 1, 2, 3, \cdots, k \tag{10}$$

Clustering techniques are considered to be one of the promising solutions for topology management for large-scale and dense wireless sensor networks. The author in [59] identifies the clustering problem as a multi-objective optimization issue and uses a GA to solve it. A wireless sensor network is partitioned into interference-free clusters, making time-slot management easier to achieve. Meanwhile, the k-means algorithm is a widely used clustering algorithm often used when clustering large-scale data sets [52].

E. COMPUTATION OFFLOADING CONSTRAINTS

The computation offloading process is complicated since there are constraints to be considered and other factors including mobility management, content cache and security and privacy. In recent years, joint optimization approaches have been proposed to minimize the overall system cost and reduce the execution and latency times [32], [60]. However, efficient scaling decisions for MEC deployments are also affected by constraints, including workload changes due to UE mobility, CPU cycles, and processor capability. For latency and execution time reduction, the execution delay of the offloaded application depends on backhaul quality and topology. For instance, a mesh topology connects the computing nodes directly, which will improve distribution, though it also requires a higher investment in the backhaul [45], [61].

III. AI FOR COMPUTATION OFFLOADING OPTIMIZATION

Generally, the optimization problem can be considered as an inherent relationship between the individual and the system. The individual stands for a point or one solution in a joint optimization function, which means the optimization aims to find the optimal solutions for an overall system function or the model solution set. For MEC computation offloading, the proposed solutions aim to find an optimal outcome in some way to improve the overall performance, maximize the computation and revenue, and consider latency and energy. Fig. 3 illustrates the categories for the algorithms widely used in computation offloading, and the rest of this section will focus on three types of AI algorithms, review the state of art, and discuss challenges.

A. EVOLUTIONARY ALGORITHMS

Evolutionary Algorithms (EA) are a ''cluster of algorithms,'' including GA [62], Multi-Objective Evolution Algorithm [63], Differential evolutionary algorithm (DE) [64], Artificial immune system [65] and others. They utilize similar mechanisms with different 'evolution modes' (functions) in their iteration processing [66].

1) GENETIC ALGORITHM

A GA is a mathematical simulation of the process of biological evolution [62]. The optimization method was developed in the 1990s to simulate the law of biological evolution [61]. A GA is a solution space consisting of a set of random solutions, which is known as the parent populations. New populations, the next generation, are formed using crossovers and mutations of the parent populations. The next step is to extract the good solutions (including the new solution) to generate further generations. The process continues until the iterative condition is reached or an optimal solution (usually a local optimal solution) is obtained. Just as with biological evolution, by inheriting excellent genes, the process aims to produce the most adapted individuals.

GA is preferred for use with large search spaces and requires substantial computational effort to find the actual Pareto [67]. In addition, when used for computation offloading optimisation, the computational cost is a key issue. Therefore, it has been proposed that GA used with particle swarm optimization and local search methods reduce the computational cost [32], [68], [69]. As GAs have improved global search performance for multi-object optimization, GA is frequently used for decision strategy analysis in computation offloading [29], [70], [71]. For example, the author in [70] proposed computation offloading methods based on Non-dominated Sorting GA III (NSGA-III). GA is utilized to find the optimal global solution for the schedules of concurrent workflow in cloud-edge computing. This activity defines the computing tasks in each schedule as a multi-objective optimization problem. The process considers the fitness function and constraints for the problem, and then adopt the crossover and mutation operations from the GA to create the new workflow schedule solutions. In [29] there is a proposed priority offloading mechanism to reduce the computation load; the GA is executed at the edge computing server to search for the best offloading strategy and transmission power.

2) DE DIFFERENTIAL EVOLUTIONARY ALGORITHM

The DE was developed by Storn and Price in 1996 [72]. The algorithm can be divided into two phases. First, it generates a set of uniformly distributed populations, and the next phase is named the evolution, which is to put the generated population through mutation, crossover and selection processes a number of times until the target criteria is achieved [73]. DE has an advantage for multi-dimensional problems with a fast convergence rate and can be adapted for continuous

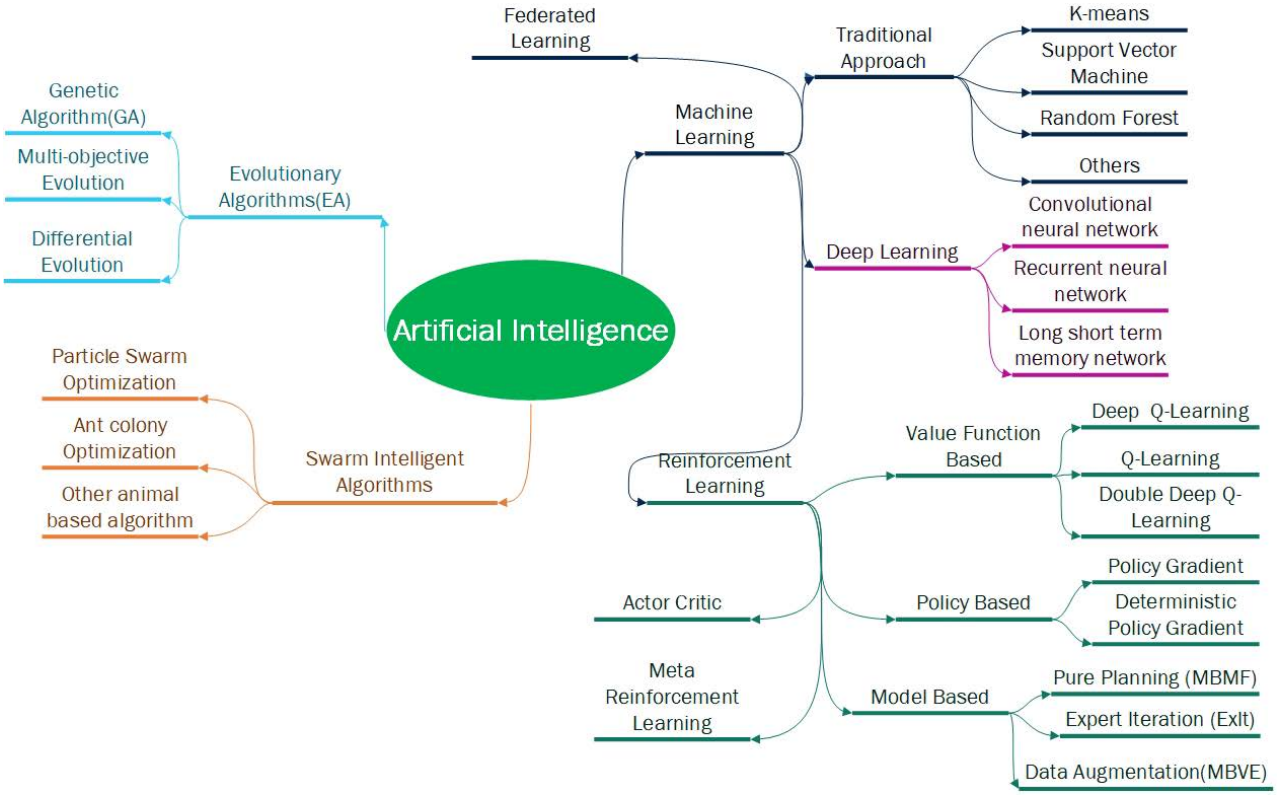

FIGURE 3. AI categories.

and discrete problems. The DE algorithm has been widely adopted for optimization problems, related to MEC utilization, such as to determine UE fine-grained offloading decisions [74], to optimize resource allocation [75] and applied to context-aware offloading strategies [76].

The authors of [74] proposed DE to determine the UE fine-grained offloading decisions by initialising the network and executing the DE algorithm, namely initialisation, mutation, crossover, and selection, to minimise the UE energy consumption. In [75] a proposed solution is provided for mining decisions. The approach adopts DE to optimise resource allocation. Since the population size is equal to the number of participating miners, it transfers the update of the number of participating miners to adjust the population size to do the optimisation. Finally, [76] proposed a context-aware offloading strategy based on DE that considers vehicle mobility, roadside unit (RSU) coverage, and vehicle priority.

3) MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM

Multi-objective evolutionary algorithm (MOEA) is a stochastic optimization technique that is similar to EA. The main difference is that the MOEA calculate the objective value for individuals at every iteration to determine the dominance relationship in the population [77]. In order to cope with scalar optimization, A MOEA Based on Decomposition (MOEA/D) was proposed by Zhang in 2007 [78], which transforms the multi-objective optimization problem into single-objective optimization by an aggregation function. In recent MEC research, MOEA/D is one of the algorithms used for multi-objective computation offloading, e.g., the problem-specific population initialization scheme uses a latency-based execution location [79] and to optimize the task offloading process in vehicular networks with RSU [80].

The algorithm in [79] focuses on the trade-off between UE execution time and energy consumption in MEC networks. MOEA/D has been proposed for use to order tasks in energy consumption based task prioritisation. The paper proposes two performance-enhancing schemes that initialize the execution location using a latency-based execution location initialization method that uses MOEA/D to compare the average computing time based on energy consumption and edge server locations. The scheme aims to decrease the energy consumption by considering the dynamic voltage and frequency scaling.

B. SWARM INTELLIGENT ALGORITHMS

SIA are inspired mainly by biological systems that adopt the collective behaviour of an organized group of animals. The SIA can generally be divided into inspect-based and animal-based [82]. SIA follow a few fundamental phases: update and move the agents by specific mechanisms to reach the defined end condition that returns the globally optimal solution [83]. The ACO and particle swarm optimization (PSO) are described in this section.

TABLE 1. EAs for intelligent offloading.

Method	Paper	Year	Objective	Proposed Techniques
GA	[81]	2020	Control the offloading proportion and reduce transmission power	Proposed a priory offloading mechanism to reduce the computation load, which changes the utility of the user device only when reaching the expectation value of time and energy consumption. The GA is applied to search the optimal offloading proportion and transmission power at the edge server side.
NSGA-III	[70]	2019	Reduce the execution time and the energy consumption	Proposed a computation offloading method based on NSGA-III to find an optimal global solution for the schedules of concurrent workflow in cloud-edge computing.
DE	[75]	2021	Maximize the total profit of all miners in MEC blockchain network	Adopt DE to optimize the resource allocation, transfer update of the number of participating miners into the adjustment of the population size, and develop an adaptive strategy.
	[76]	2021	Reduce the latency in collaborative vehicular edge computing networks	Develop a context-aware offloading strategy by considering several constraints and employing an auto-regressive integrated moving average model to predict idle computing resources according to the base station traffic in different periods.
	[74]	2021	Reduce the energy consumption of UEs and maximize mobile service providers revenue	Proposed DE algorithm to determine UE's fine-grained offloading decision and utilize Q-learning algorithm to assist the DE algorithm in selecting the proper unit service price.
MOEA	[79]	2020	Balance the execution time and the energy consumption of mobile device in MEC	Initialize the execution location using a latency-based execution location initialization method and decrease the energy consumption by considering the dynamic voltage and frequency scaling.
	[80]	2021	Optimize the task offloading process in terms of latency and energy objectives	Utilize the fog nodes for task execution and all vehicles in the system for tasks transmission.

1) PARTICLE SWARM OPTIMIZATION

PSO was firstly proposed by Kennedy and Eberhart in 1995 [84] as a population-based globalized search algorithm that mimics a swarm (cognitive and social behaviour). As aforementioned, SIA does not simulate natural phenomena like EA but instead uses the characteristics of swarms, and the key difference between PSO and evolving populations is how new samples are generated. GA's new samples are selected by some recombination of group solutions, which may then replace members of the population. And the PSO's new samples are generated by perturbation of existing solutions [85]. PSO starts from a random solution, which generates a population of particles; each particle has a memory of its moving path and an adaptable velocity for moving in the search space in each iteration. The movement is an aggregated acceleration toward the best previously visited position. Finally, particle fitness is evaluated and the optimal solution occurs after one or more iterations [86]. PSO is more straightforward than the GA, and it does not have operations like the GA crossover and mutation. Instead, PSO follows the current optimal value to find the optimal global value. PSO has been adopted widely for computation offloading decision making, e.g., as the decision strategy to reduce total computing overhead of UE [87], optimizing the time required for calculation tasks in an IoT environment [88], and to minimize the total energy consumed by UE and edge servers [32].

The author of [87] aims to minimize the total UE computing overhead when computation offloading. The problem is divided into two sub-problems. The resource allocation is defined as a convex problem, and the offloading decision is solved using Binary PSO (BPSO). The BPSO is used to solve the offloading decision problem by considering a few constraints defined by the author. Each candidate solution in the sub-problem, i.e., the offloading decision, and the offloading place, is constructed according to a single position for each particle and its velocity. [32] proposes a partial computation offloading method using joint GA Simulated-Annealing (SA) and PSO. This approach jointly optimizes the task offloading ratio, CPU speeds, channel bandwidth availability, CPU cycles and memory in edge servers. PSO is applied to update the velocities of the particles in this case.

2) ANT COLONY OPTIMIZATION

ACO is a meta-heuristic swarm intelligence approach proposed by Colorni in 1991 [89] and further improved by Dorigo in 1996 [90]. The main idea is based on the effect of pheromones on ants. The process is to derive a finite set of solution components, and by iterating the pre-defined pheromone values to assign the higher pheromone values to the better solution components probabilistically. It is more like a weighted random function. The weight is calculated using existing pheromones [91]. ACO has been applied to maximize the profits of the edge service provider [92], and to execute an offloading strategy based on two array signal processing schemes [93]. The author in [93] proposed an intelligent offloading strategy framework for MEC networks assisted by using array signal processing at the user level. The strategy is used in the ACO algorithm, where the ants randomly visit the computational access points (CAPs) to achieve optimization. The proposed ACO algorithm is utilized for scheduling in the computation offloading MEC networks. The optimization goal is to convert the shortest path to the minimal cost, each 'ant' matches a task with a given CAP, and it will leave a 'pheromone'. Then, the ant colony will make the decision based on the pheromones to achieve the optimal decision strategy for computation offloading.

TABLE 2. SIAs for intelligent offloading.

Method	Paper	Year	Objective	Proposed Techniques
BPSO	[87]	2020	Minimize the total computing overhead of mobile devices	Proposed an algorithm using a meta-heuristic based on PSO to jointly optimize resource allocation and computation-offloading decisions. Because the offloading decision and location are discrete values, using a binary version of PSO (BPSO) which can update each particle position occurs. The modified version of the BPSO can avoid local optima more effectively than the original version.
PSO	[32]	2020	Minimize the total energy consumed by mobile devices and edge servers	Proposed a partial computation offloading method based on simulated-annealing and PSO jointly optimizing the offloading ratio of tasks, CPU speed, bandwidth of available channels, CPU cycles and memories in edge servers.
ACO	[92]	2020	Minimize the total energy consumed by mobile devices and edge servers	A partial computation offloading method based on Simulated- annealing and Particle swarm optimization jointly optimizing offloading ratio of tasks, CPU speeds bandwidth of available channels CPU cycles and memories in edge servers.
	[93]	2020	Minimizing the system cost and latency	An intelligent offloading strategy to enhance the MEC network performance by utilizing array signal processing at the user level. Using 'ants' randomly visit the computational access points to achieve 'short path', which is converted to the 'minimal cost' in this situation.
PSO, ACO	[92]	2020	Balance the IoT task load over the fog nodes by considering the lateny and cost	Test two different scheduling algorithms, ACO and PSO, with a simulation.

TABLE 3. ML for intelligent offloading.

Method	Paper	Year	Objective	Proposed Techniques
DL	[94]	2019	Partial offloading scheme mainly focus on Power consumption	Proposed an offloading decision method based on deep supervised learning. The cost, which is optimized in this algorithm, involve the energy consumption, network condition, and CPU workload. DNN is trained with the state vector. The optimal offloading decision policy is used for the future decisions.
	[95]	2020	Maximizes the cellular access point and vehicle throughput	Extended an SDN-controller with DL to learn a dynamic V2X system and carried out optimum offloading.
	[96]	2021	Achieve better convergence performance, reduce computation time	Proposed an algorithm to exploit the action state and generate multiple candidate offloading decisions. Thus enhancing the convergence of deep learning.
RL	[97]	2019	Reduce the total energy consumption and the delay	Proposed a Q-learning based algorithm for partial offloading. The state point Q-value in this Q-learning algorithm is continuously iterated and trained to reach the optimal offload strategy. The rewards function includes a series of metrics of MEC performance, such as Channel state and Computing power.
	[98]	2020	Reduce the computation delay and improve the channel access success rate	multi-agent reinforcement learning to derive the optimal offloading policy. The user device plays the role of an agent that observes channels condition to determine offloading policy.
	[99]	2020	Increase the CPU utilization and reduces the execution time	Using long short-term memory model (LSTM) to estimate the future workload and reinforcement learning technique to make an appropriate scaling decision.
	[100]	2020	Optimize offloading and resources allocation jointly	The proposed algorithm is based on multi-agent Q-learning algorithm to get optimal task offloading decision. LSTM is used to predict task popularity.
DRL	[101]	2019	Reduce the total energy consumption and the delay	Modeled all possible solutions as state spaces and the movement among different states as actions, and adopted the DQN approach to find the optimal solution.
	[102]	2019	Decrease the computation time	Proposed a DRL-based Online Offloading approach that learns the binary offloading decisions from the experience to decrease the computation time.
	[103]	2020	Reduce the Power consumption	Proposed a deep learning based smart decision-making algorithm to promise the accuracy and energy consumption of UEs.
	[104]	2021	Decrease task completion time and energy consumption	Proposed an agent based on DDPG, which runs on the mobile device to make dynamic decisions on partial task offloading.
Meta RL	[105]	2021	Optimize latency and fast adaptive	Proposed an algorithm with two loops of training mode, inner loop training on the UE for the task specific policy and outer loop training on the MEC host for the meta policy.

C. MACHINE LEARNING

ML is the technique lying at the intersection of computer science and statistics, one of the most rapidly growing technologies in AI fields [106]. An offloading decision in an MEC environment must be carried out quickly. The ML-based offloading is a promising method that automatically maps a given system settings to arrive at the best offloading decision. ML algorithms vary greatly and can be categorized differently. The three general ML algorithms are categorized as: Supervised Learning (SL), Unsupervised Learning (UL), and Reinforcement Learning (RL). ML can be broadly defined as computation methods learning from experience collected data, a dataset, to improve performance or to make accurate predictions without manual manipulations [107]. In essence, ML allows users to feed a computer algorithm large amounts of data, and then let the computer analyze the data and make data-driven recommendations and decisions based solely on the input data. If the algorithm identifies any corrections,

it integrates the corrections to improve future decisions. Deep learning (DL) is an ML sub-field. The commonality between the two is that the rules are summarized through algorithms from a limited sample of a specific type of task, and can be applied to new unknown data. DL is more inclined to automatically learn effective features from raw data through algorithms. Another ML sub-field RL, which is used to train an offloading model through online interaction between a learning agent and the environment. This section will introduce DL and RL and consider the relative, and combination scenarios, e.g., Deep Reinforcement Learning.

1) DEEP LEARNING

DL represents a learning method that uses raw data and a multi-layer neural network without any manual set. The network consists of the input layer, hidden layer and output layer. Each neuron has a non-linear transformation function. Through the superposition of the functions of each layer, the output of the DL network approximates the target output infinitely [81]. The mainstream DL network structures are convolutional neural network (CNN), recurrent neural network (RNN), deep neural network (DNN), and the DL frameworks include MXnet, TensorFlow, and Caffe. DL can be applied to MEC by extracting selected features from mobile cellular networks and training the DL model on the MEC server. The trained model can be used as a decision-making engine for computation offloading [108].

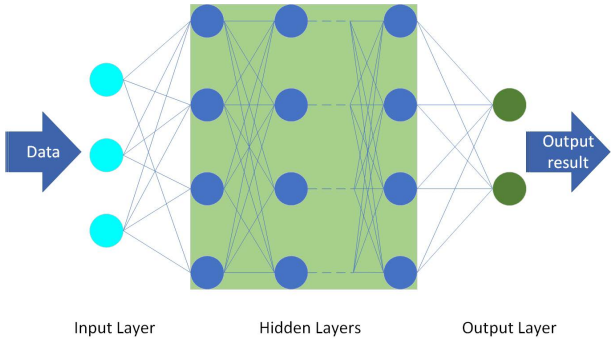

FIGURE 4. Deep Learning [16].

As shown in Figure 4, the input layer of the neural network obtains the input data from the UE, MEC and the network. After considering the number of neurons required by the requested resources and its corresponding utility value to train, the trained network calculates the required labels for the input data. The hidden layer is used to find relationships and links between output data and input data. The output layer takes the desired outcomes as labels during training. Increasing the number of hidden layers can improve performance and permit models of complex decision boundaries and to learn patterns in the data [110]. The author of [96] proposed methods to search the local space to achieve an offloading decision. The method can be used to exploit the action state

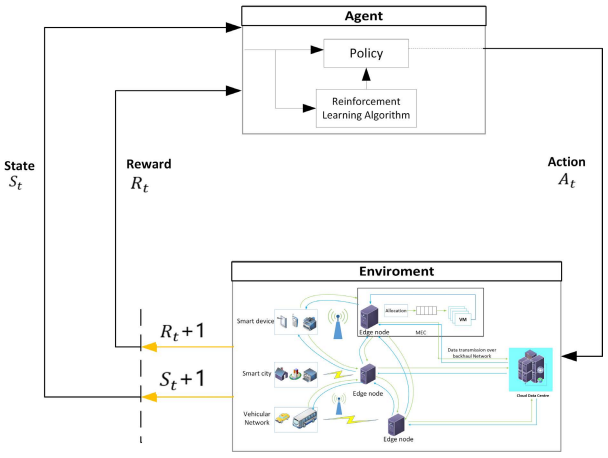

FIGURE 5. Reinforcement Learning [109].

and generate candidate offloading decisions, thus enhancing DL convergence.

2) REINFORCEMENT LEARNING

RL is a learning method that is based on the agent interacting with the environment to choose the optimal actions [111]. Specifically, the RL model is to select the action for each state of the system to maximize cumulative reward in the long term. Reinforcement learning is not learning from the data; instead, it has to learn from its own experience. Markov Decision Process (MDP) is one of the standard techniques used in RL models. The MDP system consists of four parts: state, action, transition probability and rewards. The Bellman equation [112] is derived as:

$$V(s) = R(s) + \gamma \sum_{s' \in S} P\left(s' \mid s\right) V\left(s'\right) \tag{11}$$

where R is the immediate reward, γ is a discount factor, $P\left(s' \mid s\right)$ represents the transition probability from state s to the next state s' and V is the expected total rewards. The aim of MDP is to find an optimal policy that can maximize the sum of rewards. Regarding task offloading, the MEC system can be seen to be in a state at each time point of observation, the agent will take action based on the computation offloading policy. The author in [113] has formulated the offloading problem using MDP, which maximizes the utility of the offloading system under a task delay constraint. The problem of making an offloading decision for UE in formulated using MDP in [114], and using a variable time step learning method to train the model in order to ensure the approach only makes offloading decisions when UE generate a task. The author of [115] models the process of solving an optimal computation offloading policy into a Markov decision process, where the computational offloading policy is based on task queue state and channel state between the UE and BS. To address the sharing of limited resource in an MEC system, [116] has formulated the computation offloading problem as a multi-agent MDP, and proposed a distributed learning framework which

allowed each mobile terminal to learn an optimal computation offloading policy from the interactions with the MEC system independently, and transforming the multi-agent MDP into a single-agent MDP.

In a practical network, the system model for the transition probability that can be used to solve MDP is unknown, so Q-Learning can be used as a feasible method to achieve an optimal policy. Q-Learning is a selector policy mode, which uses a traditional algorithm to create q-tables and helps the agent find the outcome.

To apply Q-Learning in MEC, the first step is to identify the actions, states, and reward functions. Then, based on exploration and exploitation, Q-Learning can update the action-value function by observing the feedback from system states and actions carried out. For Q-Learning, the agent chooses an action a on the state s using the greedy method. When the system changes to the next state s', the agent gets the reward R. For each step, the agent computes and stores $Q(s, a)$ in a q-table. This value can be regarded as a long-term reward through constant interaction with the environment [117].

$$Q(s,a) = Q(s,a) + \alpha \left[R(s,a) + \gamma \max Q\left(s', a'\right) - Q(s,a) \right] \tag{12}$$

As illustrated in Figure 3, the RL can be classified in three ways: value-based, policy-based and model-based. Value-based is to optimize the value function. The value function regards the maximum expected future rewards that the agent can expect in each state. Policy-based is optimizing the strategy function directly, while ignoring the value function. The strategy function evaluates an agent's performance at a particular point in time, and can be categorized as: deterministic strategy and random strategy. Model-based is to create a suitable model for the specific environment [118].

The most significant difference between supervised and unsupervised learning is that there is no dataset for the agent to learn when utilizing RL. The environment is constantly changing, and the RL agent has to make decisions about a series of actions in the changing environment rather than a single action decision and RL is updating policy through constant interaction with the environment.

3) DEEP REINFORCEMENT LEARNING

DL provides a more powerful prediction model and often produces good predictions, and RL provides a faster learning mechanism with adaptability to the varying environment. DL enables RL to scale to more complex problems [119], [120], which leads to the new field that was named DRL. A neural network is integrated into RL to approximate the q-values. The actor-critic method is a DL and RL integration approach. Therefore, the current mainstream policy gradient method uses the Actor-Critic (AC) framework. Actor models are used to provide the strategy, while critic models provide the value function. DRL and its related sub-fields have been widely used in MEC.

Deep Q-Network (DQN) is a typical value-based approach, and it was proposed in 2015 [121]. This approach acquires unlimited training samples through Q-Learning, and then trains the neural network. There are two key DQN steps. The first is Experience Replay, which stores the data obtained from an exploration of the system environment and then randomly samples to update the parameters of the deep neural network. The second is TargetNet, where the target q-value will remain unchanged for a period of time, which reduces the correlation between the current q-value and the target q-value to a certain extent and improves the stability of the algorithm.

For policy-based DRL, AC is a general name of a class of algorithms that can be divided into two parts: actor and critic. The actor is responsible for making decisions, used to improve the current policy. The critic is responsible for telling the actor the validity of the decision, which is used to evaluate the current state. In addition, the Deep Deterministic Policy Gradient (DDPG) [122] algorithm combines the advantages of both AC architecture and deterministic strategy. DDPG can operate over continuous action spaces and readily converges. In [123], the author proposed a resource allocation algorithm based DDPG to effectively solve the continuous power distribution. DDPG combines Deterministic Policy Gradient (DPG), uses an actor network to generate continuous behaviour, and uses a critic network to evaluate current actions, which improves convergence performance.

4) ML FOR COMPUTATIONAL OFFLOADING

For a stochastic and constantly varying MEC environment, the ML-based approach has been widely used in recent years. [94] is a DL case applied to MEC offloading, which generates data to train the DNN. Variables are selected, e.g., the remaining UE energy, local and cloud resources, the amount of data to transfer, and communication latency. The cost function will then consider the parameters in the decision-making process for computational offloading. The author of [103] proposed an online computation offloading algorithm based on DRL that is used to assign the appropriate sub-carrier to the UEs in the case of remote computation mode. An online computation offloading based on DRL is proposed in [102] that can be used for binary offloading decision making and resource allocation in wireless powered MEC networks. The meta RL model (MRL) is applied to MEC in [105]. MRL is a meta learning task based on RL techniques. There are two loops of training: the ''inner loop'' training on the UE for the task-specific policy and ''outer loop'' training on the MEC host for the meta policy.

IV. FUTURE CHALLENGES AND OPEN ISSUES

Research is ongoing into MEC and industry is in the early stages of deploying MEC. This section will briefly outline the challenges and open issues based on current research. There are three research directions: AI and ML, services and applications, security and privacy.

A. ARTIFICIAL INTELLIGENT AND MACHINE LEARNING

1) INTELLIGENT OPTIMIZATION METHODS

MEC computation offloading is a multi-objective optimization problem that needs to achieve outcomes for specific scenarios. The current intelligent optimization methods all have challenges in terms of the convergence, constraint-handling, and dimensionality [42]. Therefore, it is necessary to consider hybrid solutions for joint optimization to deal with the dynamic multi-access heterogeneous networks.

2) EDGE INTELLIGENCE

The concept of edge intelligence is starting to draw attention; enabling UE to run the pre-trained DL model from the network edge locally [124]. Federated learning (FL) is promising distributed DNN training technique that can be used to improve privacy. The author of [125] proposed leaving the raw data distributed on UE and training a shared model on the edge server by aggregating locally computed updates. Therefore, FL is a feasible future way to apply edge intelligence which enhances MEC operation.

3) COMPUTATION-AWARE NETWORKING TECHNIQUES

Computation resource awareness and a resource aware edge AI model can be used to improve transmission efficiency across the different edge nodes. In addition, the UE mobility prediction can provide efficient and stable computation offloading scheduling in some scenarios. The author of [126] proposed a weighted Markov prediction model to separately predict the trajectory of pre-classified users.

B. SERVICES AND APPLICATIONS

1) EFFICIENT SERVICE DISCOVERY PROTOCOLS

The distributed nature of MEC means that edge nodes may run different AI models and carry out different AI tasks. Therefore, it is essential to design an efficient service discovery protocol for fast identification. It can be difficult to find an optimal MEC node if the management system cannot dynamically locate UE and carry out task offloading scheduling tasks due to resource or other constraints. A new framework called "Named Function as a Service (NFaaS)" [127] was developed to extend the named data networking architecture to support function execution in the network by available resources. This approach provides a new avenue for future research.

2) APPLICATION PARTITIONING

Partitioning granularity is a challenge as some of the task execution requires consistent runtime environments [38]. Research into the consistency of the MEC environment remains ongoing.

C. SECURITY AND PRIVACY

Security and privacy are always critical issues in any network. Therefore, the security and privacy of data and edge infrastructure is important for computation offloading. Security and privacy challenges remain open for security configuration, threat detection, threat remediation and network verification. Software-defined segmentation and trust management authentication are feasible options to reduce the attack surface from the server side [36]. It is impractical to run SSL/TLS protocols on the UE in some scenarios [128] so a challenge remains to identify lightweight security protocols for the MEC environment that can be used for user authentication, access control, key and identity management. The trade-off between task computation for data encryption or that required to utilize security certificates should also be considered in the application partitioning algorithm.

V. CONCLUSION

Computation offloading is one of the significant directions being pursued that could enhance performance and the end user experience. The computation offloading decision strategy to be employed for the dynamic MEC environment should include various factors, e.g., resource allocation and content caching. Furthermore, the joint optimization approaches should consider working with mobility management synchronization and service migration. This article introduced computation offloading an MEC environment, discussed concepts, intelligent computation offloading optimization, and outlined future challenges and open issues.

REFERENCES

[1] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 2017.

[2] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 752–764, Jan. 2018.

[3] X. Deng, J. Li, E. Liu, and H. Zhang, "Task allocation algorithm and optimization model on edge collaboration," *J. Syst. Archit.*, vol. 110, Nov. 2020, Art. no. 101778.

[4] P. Cong, J. Zhou, L. Li, K. Cao, T. Wei, and K. Li, "A survey of hierarchical energy optimization for mobile edge computing: A perspective from end devices to the cloud," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–44, Mar. 2021.

[5] C. Wu, Q. Peng, Y. Xia, Y. Ma, W. Zheng, H. Xie, S. Pang, F. Li, X. Fu, X. Li, and W. Liu, "Online user allocation in mobile edge computing environments: A decentralized reactive approach," *J. Syst. Archit.*, vol. 113, Feb. 2021, Art. no. 101904.

[6] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[7] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart., 2018.

[8] P. Ameigeiras, J. J. Ramos-Munoz, L. Schumacher, J. Prados-Garzon, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Link-level access cloud architecture design based on SDN for 5G networks," *IEEE Netw.*, vol. 29, no. 2, pp. 24–31, Mar./Apr. 2015.

[9] 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.

[10] 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.

[11] K. Ha, Y. Abe, T. Eiszler, Z. Chen, W. Hu, B. Amos, R. Upadhyaya, P. Pillai, and M. Satyanarayanan, "You can teach elephants to dance: Agile VM handoff for edge computing," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 1–14.

[12] L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via Docker container migration," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 1–13.

[13] A. Islam, A. Debnath, M. Ghose, and S. Chakraborty, "A survey on task offloading in multi-access edge computing," *J. Syst. Archit.*, vol. 118, Sep. 2021, Art. no. 102225.

[14] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[15] 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. 2015.

[16] B. Cao, L. Zhang, Y. Li, D. Feng, and W. Cao, "Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 56–62, Mar. 2019.

[17] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, and I. Humar, "A dynamic service migration mechanism in edge cognitive computing," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–15, May 2019.

[18] 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.

[19] G. Carvalho, B. Cabral, V. Pereira, and J. Bernardino, "Computation offloading in edge computing environments using artificial intelligence techniques," *Eng. Appl. Artif. Intell.*, vol. 95, Oct. 2020, Art. no. 103840.

[20] N. Primeau, R. Falcon, R. Abielmona, and E. M. Petriu, "A review of computational intelligence techniques in wireless sensor and actuator networks," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2822–2854, 4th Quart., 2018.

[21] Y. Tao, X. Wang, X. Xu, and Y. Chen, "Dynamic resource allocation algorithm for container-based service computing," in *Proc. IEEE 13th Int. Symp. Auto. Decentralized Syst. (ISADS)*, Mar. 2017, pp. 61–67.

[22] Y. Liu, J. Zhao, Z. Xiong, D. Niyato, C. Yuen, C. Pan, and B. Huang, "Intelligent reflecting surface meets mobile edge computing: Enhancing wireless communications for computation offloading," 2020, *arXiv:2001.07449*.

[23] Z. Xiao, X. Dai, H. Jiang, and D. Wang, "Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2038–2052, Mar. 2020.

[24] M. Asim, Y. Wang, K. Wang, and P.-Q. Huang, "A review on computational intelligence techniques in cloud and edge computing," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 6, pp. 742–763, Dec. 2020.

[25] W. Tong, A. Hussain, W. X. Bo, and S. Maharjan, "Artificial intelligence for vehicle-to-everything: A survey," *IEEE Access*, vol. 7, pp. 10823–10843, 2019.

[26] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.

[27] Z. Mohamad, A. A. Mahmoud, W. Nik, M. A. Mohamed, and M. M. Deris, "A genetic algorithm for optimal job scheduling and load balancing in cloud computing," *Int. J. Eng. Technol.*, vol. 7, no. 3, pp. 290–294, 2018.

[28] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 522–533, Jun. 2019.

[29] J. Wang, W. Wu, Z. Liao, R. S. Sherratt, G.-J. Kim, O. Alfarraj, A. Alzubi, and A. Tolba, "A probability preferred priori offloading mechanism in mobile edge computing," *IEEE Access*, vol. 8, pp. 39758–39767, 2020.

[30] T. Wang, X. Wei, C. Tang, and J. Fan, "Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints," *Peer Peer Netw. Appl.*, vol. 11, no. 4, pp. 793–807, 2018.

[31] Y. Guo, Z. Zhao, R. Zhao, S. Lai, Z. Dan, J. Xia, and L. Fan, "Intelligent offloading strategy design for relaying mobile edge computing networks," *IEEE Access*, vol. 8, pp. 35127–35135, 2020.

[32] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3774–3785, Mar. 2021.

[33] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107496.

[34] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1529–1541, Jul. 2021.

[35] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020.

[36] B. Ali, M. A. Gregory, and S. Li, "Multi-access edge computing architecture, data security and privacy: A review," *IEEE Access*, vol. 9, pp. 18706–18721, 2021.

[37] B. Liang, M. A. Gregory, and S. Li, "Multi-access edge computing fundamentals, services, enablers and challenges: A complete survey," *J. Netw. Comput. Appl.*, vol. 199, Mar. 2022, Art. no. 103308.

[38] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.

[39] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020.

[40] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[41] G. Carvalho, B. Cabral, V. Pereira, and J. Bernardino, "Computation offloading in edge computing environments using artificial intelligence techniques," *Eng. Appl. Artif. Intell.*, vol. 95, Oct. 2020, Art. no. 103840.

[42] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Generat. Comput. Syst.*, vol. 70, pp. 59–63, May 2017.

[43] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[44] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[45] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[46] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported Internet of Things," *IEEE Access*, vol. 7, pp. 69194–69201, 2019.

[47] H. Q. Le, H. Al-Shatri, and A. Klein, "Efficient resource allocation in mobile-edge computation offloading: Completion time minimization," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2513–2517.

[48] F. Wang and X. Zhang, "Dynamic interface-selection and resource allocation over heterogeneous mobile edge-computing wireless networks with energy harvesting," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 190–195.

[49] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang, "A survey on computation offloading modeling for edge computing," *J. Netw. Comput. Appl.*, vol. 169, Nov. 2020, Art. no. 102781.

[50] 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. 2015.

[51] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[52] 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.

[53] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[54] M. Deng, H. Tian, and B. Fan, "Fine-granularity based application offloading policy in cloud-enhanced small cell networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, May 2016, pp. 638–643.

[55] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2016.
[56] S. Bi and Y. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Apr. 2018.
[57] Z. Ding, P. Fan, and H. V. Poor, "Impact of non-orthogonal multiple access on the offloading of mobile edge computing," *IEEE Trans. Commun.*, vol. 67, no. 1, pp. 375–390, Jan. 2019.
[58] J. Zhu, J. Wang, Y. Huang, F. Fang, K. Navaie, and Z. Ding, "Resource allocation for hybrid NOMA MEC offloading," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4964–4977, Jul. 2020.
[59] Z. Chen, H. Lin, L. Wang, and B. Zhao, "Interference-free clustering protocol for large-scale and dense wireless sensor networks," *KSII Trans. Internet Inf. Syst. (TIIS)*, vol. 13, no. 3, pp. 1238–1259, 2019.
[60] A. Shahidinejad and M. Ghobaei-Arani, "Joint computation offloading and resource provisioning for edge-cloud computing environment: A machine learning-based approach," *Softw., Pract. Exper.*, vol. 50, no. 12, pp. 2212–2230, Dec. 2020.
[61] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.
[62] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.
[63] X. Xu, X. Zhang, X. Liu, J. Jiang, L. Qi, and M. Z. A. Bhuiyan, "Adaptive computation offloading with edge for 5G-envisioned internet of connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5213–5222, Aug. 2021.
[64] Y. Sun, C. Song, S. Yu, Y. Liu, H. Pan, and P. Zeng, "Energy-efficient task offloading based on differential evolution in edge computing system with energy harvesting," *IEEE Access*, vol. 9, pp. 16383–16391, 2021.
[65] C. A. C. Coello and N. C. Cortés, "Solving multiobjective optimization problems using an artificial immune system," *Genetic Program. Evolvable Mach.*, vol. 6, no. 2, pp. 163–190, 2005.
[66] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *Proc. Int. Conf. Global Trends Signal Process., Inf. Comput. Commun. (ICGTSPICC)*, Dec. 2016, pp. 261–265.
[67] Z. Wang and A. Sobey, "A comparative review between genetic algorithm use in composite optimisation and the state-of-the-art in evolutionary computation," *Composite Struct.*, vol. 233, Feb. 2020, Art. no. 111739.
[68] O.-K. Shahryari, H. Pedram, V. Khajehvand, and M. D. TakhtFooladi, "Energy and task completion time trade-off for task offloading in fog-enabled IoT networks," *Pervas. Mobile Comput.*, vol. 74, Jul. 2021, Art. no. 101395.
[69] P. Anusha and R. S. Balan, "Efficient power management in mobile computing with edge server offloading using multi-objective optimization," *EAI Endorsed Trans. Energy Web*, vol. 9, no. 37, Jul. 2021.
[70] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 522–533, Jun. 2019.
[71] X. Xu, Y. Li, T. Huang, Y. Xue, K. Peng, L. Qi, and W. Dou, "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," *J. Netw. Comput. Appl.*, vol. 133, pp. 75–85, May 2019.
[72] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
[73] M. Pant, H. Zaheer, L. Garcia-Hernandez, and A. Abraham, "Differential evolution: A review of more than two decades of research," *Eng. Appl. Artif. Intell.*, vol. 90, Apr. 2020, Art. no. 103479.
[74] Y. Liao, X. Qiao, Q. Yu, and Q. Liu, "Intelligent dynamic service pricing strategy for multi-user vehicle-aided MEC networks," *Future Gener. Comput. Syst.*, vol. 114, pp. 15–22, Jan. 2021.
[75] Y. Wang, C.-R. Chen, P.-Q. Huang, and K. Wang, "A new differential evolution algorithm for joint mining decision and resource allocation in a MEC-enabled wireless blockchain network," *Comput. Ind. Eng.*, vol. 155, May 2021, Art. no. 107186.
[76] Z. Jin, C. Zhang, G. Zhao, Y. Jin, and L. Zhang, "A context-aware task offloading scheme in collaborative vehicular edge computing systems," *KSII Trans. Internet Inf. Syst.*, vol. 15, no. 2, pp. 383–403, 2021.
[77] N. Gunantara, "A review of multi-objective optimization: Methods and its applications," *Cogent Eng.*, vol. 5, no. 1, Jan. 2018, Art. no. 1502242.
[78] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[79] F. Song, H. Xing, S. Luo, D. Zhan, P. Dai, and R. Qu, "A multiobjective computation offloading algorithm for mobile-edge computing," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8780–8799, Sep. 2020.
[80] S. Abdullah and A. Jabir, "A light weight multi-objective task offloading optimization for vehicular fog computing," *Iraqi J. Electr. Electron. Eng.*, vol. 17, no. 1, pp. 1–10, Jun. 2021.
[81] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[82] A. Chakraborty and A. K. Kar, *Swarm Intelligence: A Review of Algorithms*. Cham, Switzerland: Springer, 2017, pp. 475–494.
[83] L. Brezočnik, I. Fister, and V. Podgorelec, "Swarm intelligence algorithms for feature selection: A review," *Appl. Sci.*, vol. 8, no. 9, p. 1521, Sep. 2018.
[84] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE ICNN*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.
[85] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization—Part I: Background and development," *Natural Comput.*, vol. 6, no. 4, pp. 467–484, Dec. 2007.
[86] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Comput.*, vol. 1, nos. 2–3, pp. 235–306, 2002.
[87] L. N. T. Huynh, Q.-V. Pham, X.-Q. Pham, T. D. T. Nguyen, M. D. Hossain, and E.-N. Huh, "Efficient computation offloading in multi-tier multi-access edge computing systems: A particle swarm optimization approach," *Appl. Sci.*, vol. 10, no. 1, p. 203, Dec. 2019.
[88] S. Dai, M. Liwang, Y. Liu, Z. Gao, L. Huang, and X. Du, "Hybrid quantum-behaved particle swarm optimization for mobile-edge computation offloading in Internet of Things," in *Proc. Int. Conf. Mobile Ad-Hoc Sensor Netw.* Cham, Switzerland: Springer, 2017, pp. 350–364.
[89] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proc. 1st Eur. Conf. Artif. Life*, vol. 142, Dec. 1991, pp. 134–142.
[90] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.
[91] R. F. T. Neto and M. G. Filho, "Literature review regarding ant colony optimization applied to scheduling problems: Guidelines for implementation and directions for future research," *Eng. Appl. Artif. Intell.*, vol. 26, no. 1, pp. 150–161, Jan. 2013.
[92] M. K. Hussein and M. H. Mousa, "Efficient task offloading for IoT-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37191–37201, 2020.
[93] Y. Guo, Z. Zhao, R. Zhao, S. Lai, Z. Dan, J. Xia, and L. Fan, "Intelligent offloading strategy design for relaying mobile edge computing networks," *IEEE Access*, vol. 8, pp. 35127–35135, 2020.
[94] Z. Ali, L. Jiao, T. Baker, G. Abbas, Z. H. Abbas, and S. Khaf, "A deep learning approach for energy efficient computational offloading in mobile edge computing," *IEEE Access*, vol. 7, pp. 149623–149633, 2019.
[95] B. Fan, Z. He, Y. Wu, J. He, Y. Chen, and L. Jiang, "Deep learning empowered traffic offloading in intelligent software defined cellular V2X networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13328–13340, Nov. 2020.
[96] Z. Wan, X. Dong, and C. Deng, "Deep learning with enhanced convergence and its application in MEC task offloading," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.* Cham, Switzerland: Springer, 2021, pp. 361–375.
[97] J. Liu, S. Wang, J. Wang, C. Liu, and Y. Yan, "A task oriented computation offloading algorithm for intelligent vehicle network with mobile edge computing," *IEEE Access*, vol. 7, pp. 180491–180502, 2019.
[98] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6201–6213, Jul. 2020.
[99] A. Shahidinejad and M. Ghobaei-Arani, "Joint computation offloading and resource provisioning for edge-cloud computing environment: A machine learning-based approach," *Softw., Pract. Exper.*, vol. 50, no. 12, pp. 2212–2230, Dec. 2020.

[100] Z. Yang, Y. Liu, Y. Chen, and N. Al-Dhahir, "Cache-aided NOMA mobile edge computing: A reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6899–6915, Oct. 2020.

[101] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, 2018.

[102] L. Huang, S. Bi, and Y.-J.-A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[103] M. Nduwayezu, Q.-V. Pham, and W.-J. Hwang, "Online computation offloading in NOMA-based multi-access edge computing: A deep reinforcement learning approach," *IEEE Access*, vol. 8, pp. 99098–99109, 2020.

[104] J. Chen, H. Xing, Z. Xiao, L. Xu, and T. Tao, "A DRL agent for jointly optimizing computation offloading and resource allocation in MEC," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17508–17524, Dec. 2021.

[105] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 242–253, Jan. 2021.

[106] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[107] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2018.

[108] S. Yu, X. Chen, L. Yang, D. Wu, M. Bennis, and J. Zhang, "Intelligent edge: Leveraging deep imitation learning for mobile edge computation offloading," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 92–99, Feb. 2020.

[109] J. Liu, M. Ahmed, M. A. Mirza, W. U. Khan, D. Xu, J. Li, A. Aziz, and Z. Han, "RL/DRL meets vehicular task offloading using edge and vehicular cloudlet: A survey," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8315–8338, Jun. 2022.

[110] Z. Ali, S. Khaf, Z. H. Abbas, G. Abbas, F. Muhammad, and S. Kim, "A deep learning approach for mobility-aware and energy-efficient resource allocation in MEC," *IEEE Access*, vol. 8, pp. 179530–179546, 2020.

[111] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[112] Z. Ding, Y. Huang, H. Yuan, and H. Dong, "Introduction to reinforcement learning," in *Deep Reinforcement Learning*. Cham, Switzerland: Springer, 2020, pp. 47–123.

[113] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7635–7647, Oct. 2019.

[114] T. Zhang, Y.-H. Chiang, C. Borcea, and Y. Ji, "Learning-based offloading of tasks with diverse delay sensitivities for mobile edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[115] X. Zheng, M. Li, M. Tahir, Y. Chen, and M. Alam, "Stochastic computation offloading and scheduling based on mobile edge computing," *IEEE Access*, vol. 7, pp. 72247–72256, 2019.

[116] X. Chen, C. Wu, Z. Liu, N. Zhang, and Y. Ji, "Computation offloading in beyond 5G networks: A distributed learning framework and applications," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 56–62, Apr. 2021.

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

[118] C. Szepesvári, "Algorithms for reinforcement learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 4, no. 1, pp. 1–103, 2010.

[119] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[120] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," 2017, *arXiv:1708.05866*.

[121] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and S. Petersen, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[122] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[123] J. Ren and S. Xu, "DDPG based computation offloading and resource allocation for MEC systems with energy harvesting," in *Proc. IEEE 93rd Veh. Technol. Conf. (VTC-Spring)*, Apr. 2021, pp. 1–5.

[124] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[125] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[126] M. Yan, S. Li, C. A. Chan, Y. Shen, and Y. Yu, "Mobility prediction using a weighted Markov model based on mobile user classification," *Sensors J.*, vol. 21, no. 5, p. 1740, 2021.

[127] M. Król and I. Psaras, "NFaaS: Named function as a service," in *Proc. 4th ACM Conf. Inf.-Centric Netw.*, Sep. 2017, pp. 134–144.

[128] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE Trans. Rel.*, vol. 64, no. 3, pp. 1086–1097, Sep. 2015.

HENGLI JIN (Student Member, IEEE) received the bachelor's degree in electrical and electronics engineering and the master's degree in telecommunication and networks from RMIT University, in 2018 and 2020, respectively. He is currently pursuing the Ph.D. degree with the School of Engineering, RMIT University, Melbourne, Australia. His research interests include SDN networks and MEC optimization.

MARK A. GREGORY (Senior Member, IEEE) received the Ph.D. degree from RMIT University, Melbourne, Australia, in 2008. He is a fellow of the Institute of Engineers Australia, where he is an Associate Professor with the School of Engineering. In 2009, he received an Australian Learning and Teaching Council Citation for an outstanding contribution to teaching and learning. His research interests include telecommunications, network design, public policy, and technical risk. He is the General Co-Chair of ITNAC. He is the Managing Editor of an international journals *International Journal of Information, Communication Technology and Applications* (IJICTA).

SHUO LI (Member, IEEE) received the B.Eng. and Ph.D. degrees from the City University of Hong Kong, Hong Kong, SAR, in 2009 and 2014, respectively. She is a Lecturer with the School of Engineering, RMIT University, Australia. Her research interests include analysis and design of telecommunications networks and multi-access edge computing and security.