Received February 3, 2022, accepted March 16, 2022, date of publication March 21, 2022, date of current version March 30, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3160738

Dynamic Service Placement in Multi-Access Edge Computing: A Systematic Literature Review

HADI TABATABAEE MALAZI[1], **SAQIB RASOOL CHAUDHRY**[2], **AQEEL KAZMI**[3], **ANDREI PALADE**[3], **CHRISTIAN CABRERA**[4], **GARY WHITE**[3], **AND SIOBHÁN CLARKE**[3]

[1]Department of Computer Science, Maynooth University, Maynooth, Co. Kildare, Ireland
[2]Department of Computer Science, Munster Technological University, Cork, T12 P928 Ireland
[3]School of Computer Science and Statistics, Trinity College Dublin, Dublin 2, Ireland
[4]Department of Computer Science and Technology, University of Cambridge, Cambridge CB2 1TN, U.K.

Corresponding author: Hadi Tabatabaee Malazi (hadi.tabatabaee@mu.ie)

This work was supported by the Science Foundation Ireland (SFI), Enable Project, under Grant 16/SP/3804.

ABSTRACT The advent of new cloud-based applications such as mixed reality, online gaming, autonomous driving, and healthcare has introduced infrastructure management challenges to the underlying service network. Multi-access edge computing (MEC) extends the cloud computing paradigm and leverages servers near end-users at the network edge to provide a cloud-like environment. The optimum placement of services on edge servers plays a crucial role in the performance of such service-based applications. Dynamic service placement problem addresses the adaptive configuration of application services at edge servers to facilitate end-users and those devices that need to offload computation tasks. While reported approaches in the literature shed light on this problem from a particular perspective, a panoramic study of this problem reveals the research gaps in the big picture. This paper introduces the dynamic service placement problem and outline its relations with other problems such as task scheduling, resource management, and caching at the edge. We also present a systematic literature review of existing dynamic service placement methods for MEC environments from networking, middleware, applications, and evaluation perspectives. In the first step, we review different MEC architectures and their enabling technologies from a networking point of view. We also introduce different cache deployment solutions in network architectures and discuss their design considerations. The second step investigates dynamic service placement methods from a middleware viewpoint. We review different service packaging technologies and discuss their trade-offs. We also survey the methods and identify eight research directions that researchers follow. Our study categorises the research objectives into six main classes, proposing a taxonomy of design objectives for the dynamic service placement problem. We also investigate the reported methods and devise a solutions taxonomy comprising six criteria. In the third step, we concentrate on the application layer and introduce the applications that can take advantage of dynamic service placement. The fourth step investigates evaluation environments used to validate the solutions, including simulators and testbeds. We introduce real-world datasets such as edge server locations, mobility traces, and service requests used to evaluate the methods. We compile a list of open issues and challenges categorised by various viewpoints in the last step.

INDEX TERMS Mobile edge computing, decentralised cloud, MEC server, service caching, service offloading, computational offloading, service deployment, resource management, service orchestration.

I. INTRODUCTION

The ever-growing number of mobile devices (e.g., smartphones, wearable gadgets, drones and connected vehicles) broadens the diversity of cloud-based applications within domains such as smart cities, industry 4.0, healthcare and cooperative driving. Cisco predicted nearly 300 billion mobile applications to be downloaded by 2023 [1]. Along with the thriving range of new and diverse services, the expectations towards immersive quality of experience (QoE) are also increasing, putting centralised mobile cloud computing environments under pressure since linearly increasing computational resources in cloud computing ecosystems cannot fulfil this demand. Additionally, the Ericsson Mobility

The associate editor coordinating the review of this manuscript and approving it for publication was Petros Nicopolitidis.

Report [2] forecasts that global total mobile data traffic will increase from 33 EB per month at the end of 2019 to 164EB per month in 2025. The related performance requirements include the support of up to 1000 times higher data volumes, data rates up to 10 Gb/s, very low service level latency below 5ms, ubiquitous communicating things and mass connectivity supporting 300,000 devices within a single cell, ultra-high reliability of 99.999% (i.e., five-nine availability) and reduced energy consumption by 90% [3], [4]. As a result, core network limitations will challenge the conventional centralised mobile cloud computing models since mobile applications use up-links to connect to a central cloud data centre via a backhaul network. One potential solution to these challenging requirements is to transfer computations to centralised clouds, which can be burdened by many issues, such as network congestion and privacy policies. This has driven the birth of multi-access edge computing (MEC).

A. MEC ENVIRONMENT

Prior to MEC, there have been some similar computing concepts, for example, mobile cloud computing (MCC), cloudlet, and Fog computing. MCC combines cloud computing, mobile computing, and wireless communication networks, thus enabling developers and service providers to support more complex applications by moving the computing capabilities and data storage away from mobile devices and into the cloud [5]. However, MCC suffers from considerable disadvantages, e.g., low scalability, high latency, privacy and security issues, and extreme burden on limited bandwidth. As the first edge computing concept, cloudlet [6] refers to a trusted and resource-rich computer or a cluster of computers located in a strategic location at the network edge and well connected to the Internet. The primary purpose of cloudlet is to extend cloud computing to the network edge and support resource-constrained mobile users in running resource-intensive and interactive applications. The WiFi connection between users and cloudlets can be a severe drawback. In particular, users cannot access cloudlets in the long-distance and use both WiFi and cellular connection simultaneously [7], i.e., users have to switch between the mobile network and WiFi when they use cloudlet services. Fog computing, a term put forward by Cisco in 2012, refers to the extension of cloud computing from the core to the network edge, reducing the amount of data needed to transfer to the central cloud [8]. Fog computing plays an essential role in many use cases and applications [8], e.g., smart cities, connected vehicles, smart grid, wireless sensor and actuator networks, smart buildings, and decentralised smart building control. However, a Fog node cannot act as a self-managed cloud data centre and needs the support of the cloud. The cloudlet and Fog computing are similar in that cloudlets, and Fog nodes are not integrated into the mobile network architecture. Thus Fog nodes and cloudlets are commonly deployed and owned by private enterprises.

The mobile edge computing industry specification group (MEC ISG) of the European telecommunications standards institute (ETSI) initiated the MEC concept in 2014. As a complement of the C-RAN architecture, MEC aims to unite the telecommunication and IT cloud services to provide the cloud-computing capabilities within radio access networks in the close vicinity of mobile users [9]. To reap additional benefits of MEC with heterogeneous access technologies, e.g., 4G, 5G, WiFi, and fixed connection, ETSI ISG officially changed the name of mobile edge computing to mean multi-access edge computing. After this scope expansion, edge servers (MEC servers) can be deployed by the network operators at various locations within RAN and/or collocated with different elements of the network edge, such as base-stations (eNB in 4G and gNB in 5G), optical network units, radio network controller sites, and WiFi access points. This transforms the edge to better facilitate communication functionalities and computation, caching, and control services. MEC can offer an ultra-low latency environment using intermediate edge servers that reduce user-cloud data exchange and service access latency, enabling service providers to deliver a higher quality of service (QoS). The edge servers also provide multi-tenancy and multi-service facilities compatible with the micro-service architectural style. For example, consider a mobile user who sends an image to a cloud-based data centre requesting a foreign language translation [10]. Although the user may benefit from extensive computation resources of the cloud, the long distance between the user and the data centre will result in a noticeable delay as well as consuming backhaul network bandwidth. Placing such services in edge servers close to users will reduce service response time, distribute the processing load of the cloud, and reduce core network traffic. Two MEC environment characteristics suggest that service placement should be dynamic. First, edge servers cannot host all possible services for their resource constraints. Second, demand patterns are non-stationary/not known apriori, which means demand is subject to change over time and space as the locations of mobile users change [11]. Service placement decisions should be dynamic and change over time because both demand and consumer proximity to server locations change [12].

B. DYNAMIC SERVICE PLACEMENT

Dynamic service placement refers to an adaptive configuration of services, including libraries and data stores, on edge servers to facilitate mobile users offloading their tasks to an appropriate edge server [13]. Generally, each service can be cached in multiple edge servers, and each server can host multiple services up to its resource limitations. Besides, all services are placed in the cloud [14]. When a service request from a mobile user arrives at an edge server, the server will respond to the user immediately if it has cached the service. Otherwise, it will forward the request to a nearby edge server or the cloud. An edge server can also download the whole service and process it locally [15].

The dynamic service placement problem can be divided into three sub-problems. The first is service caching, which focuses on deciding which services should be placed in a particular edge server. The second is resource allocation, which deals with managing the available resources in an edge server. The third is task forwarding (offloading), in which a decision has to be made as to which one of the edge servers or the cloud is responsible for computing the requested service.

Dynamic service placement introduces several challenges. First, the spatio-temporal demand pattern for a service is unknown since users are mobile and their interests change, while the switching cost of activating and deactivating services cannot be ignored. Second, the resources at an edge server are limited compared to those in data centres. Third, application services are heterogeneous along multiple dimensions. For instance, applications may have different QoS requirements. The sizes of application services (as containers or virtual machines) are unlikely to be equal. Moreover, differing input/output data sizes for services can affect the placement performance. Finally, the edge servers are managed by different administrative domains, following different goals and possible conflicts of interest.

C. RELATED SURVEYS

Several related survey papers have been published in recent years. We review these surveys and classify them into three groups. The first group includes the works investigating service placement problems in distributed computing paradigms. We summarise the key enhancements of our paper compared to these surveys. The second group consists of works that review related problems such as computational offloading, task scheduling, service migration, VM management, resource management (allocation and provisioning), content caching, and security in an edge computing paradigm. Finally, the last group is dedicated to a MEC environment and covers a more extensive scope by studying a wide range of issues.

D. OUR CONTRIBUTIONS

This paper presents a systematic literature review of dynamic service placement in a MEC environment. We address the following five research questions by adopting the layered conceptual model of distributed systems (i.e., network, middleware, and application layers).

RQ1: What is the effect of networking technologies on dynamic service placement solutions? We review the literature from the network layer perspective investigate different aspects of MEC architecture. We also study the enabling technologies, including software defined networking, network function virtualisation (NFV), information-centric networking (ICN), and network slicing. Finally, we explore different cache deployment solutions in 5G MEC, such as heterogeneous networks (HetNet), cloud radio access networks (C-RAN), and heterogeneous cloud radio access networks.

RQ2: What are the current research directions in dynamic service placement? We identify eight research directions that influence the system model of dynamic service placement methods and introduce different aspects of these directions.

RQ3: What is the taxonomy of dynamic service placement methods? We devise the taxonomy of design objectives based on the six classes of design goals that we identify. Besides, we propose a taxonomy of solutions including six different criteria and review the works based on them.

RQ4: Which category of applications can benefit from dynamic service placement? We review the QoS requirements of applications. Then, survey several application scenarios that can benefit from dynamic service placement.

RQ5: What are the current research validation/evaluation methods? We introduce the performance evaluation metrics used, then analyse different evaluation environments used in dynamic service placement. We also survey some of the real-world data used in the literature.

Finally, we describe several open issues and challenges from the network, middleware, application layer viewpoints, and evaluation methods.

E. SCOPE AND DOMAIN

This systematic literature review focuses on research published on dynamic service placement in MEC environments, emphasising MEC (edge) servers. The following topics are included/excluded.

Included subjects:

- Service caching in edge servers.
- Edge server resource allocation.
- Service offloading: Computational offloading is a widely-used term and has rich literature reviews. In this work, we concentrate on an area that has received less attention by reviewing the works that support single application multiple users (SAMU) as well as multiple application multiple users MAMU) based on the proposed tenancy classification of distributed systems [16].

Excluded subjects:

- User device to user device offloading: We do not cover device to device offloading and refer interested readers to [17]–[21], which provide a broad insight into computational offloading.
- Static service/application placement: The demand pattern of services in the MEC environment is non-stationary and not known apriori. Additionally, the ever-changing network conditions and limited edge server resources can only be addressed in dynamic service placement. Hence, we excluded static service placement, and application deployment works.
- Content caching: Service placement and content caching have several distinguished differences. First, content caching mainly concerns a storage resource, while service placement considers the storage and focuses on other computing resources such as processing and memory. The edge server that hosts a service has a constraint on how many service requests it can serve [13]. Second, requesting a service consumes non-negligible energy, whereas accessing content consumes a negligible amount of energy [15]. Third, dynamic service

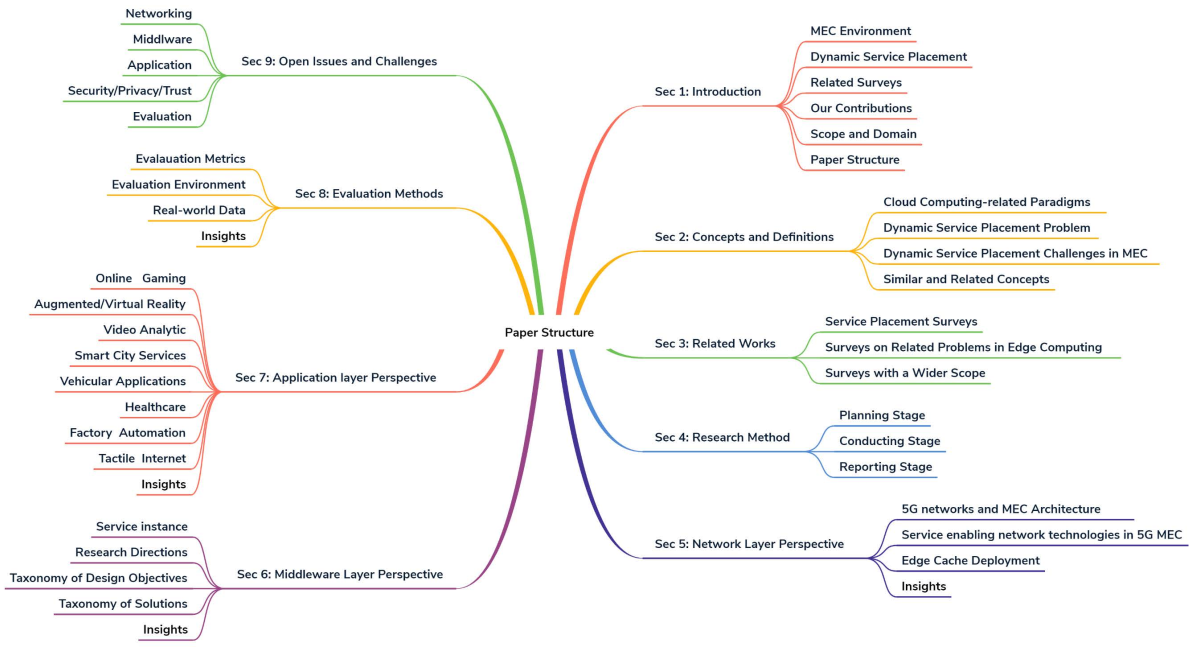
Paper Structure
Sec 1: Introduction
MEC Environment
Dynamic Service Placement
Related Surveys
Our Contributions
Scope and Domain
Paper Structure
Sec 2: Concepts and Definitions
Cloud Computing-related Paradigms
Dynamic Service Placement Problem
Dynamic Service Placement Challenges in MEC
Similar and Related Concepts
Sec 3: Related Works
Service Placement Surveys
Surveys on Related Problems in Edge Computing
Surveys with a Wider Scope
Sec 4: Research Method
Planning Stage
Conducting Stage
Reporting Stage
Sec 5: Network Layer Perspective
5G networks and MEC Architecture
Service enabling network technologies in 5G MEC
Edge Cache Deployment
Insights
Sec 6: Middleware Layer Perspective
Service instance
Research Directions
Taxonomy of Design Objectives
Taxonomy of Solutions
Insights
Sec 7: Application layer Perspective
Online Gaming
Augmented/Virtual Reality
Video Analytic
Smart City Services
Vehicular Applications
Healthcare
Factory Automation
Tactile Internet
Insights
Sec 8: Evaluation Methods
Evalauation Metrics
Evaluation Environment
Real-world Data
Insights
Sec 9: Open Issues and Challenges
Networking
Middlware
Application
Security/Privacy/Trust
Evaluation

FIGURE 1. An overview of the paper structure.

placement methods aim at improving different service quality metrics. Fourth, the cache update cost is usually omitted in content caching methods [15], which is not the case in dynamic service placement. Finally, task sharing among servers [22], [23] and service chaining make dynamic service placement a distinctive problem. The costs of service access and cache update coupled with the interplay between the available services in the cache and the computing capacity at the base stations make the service caching and task-assignment policies more challenging to design. Yao *et al.* [24] provide a thorough analysis of the content caching problem on mobile edge computing.

F. PAPER STRUCTURE

The rest of this paper is organised as follows: The next section introduces some of the main decentralised computing paradigms, preliminary concepts and defines the dynamic service placement problem. In Section III, similar surveys are reviewed, and differences are highlighted. The systematic literature review method used is described in Section IV, while dynamic service placement is discussed from the network layer, middleware layer, and application layer perspectives in Sections V, VI, and VII, respectively. Section VIII has dedicated to evaluation methods, and Section IX describes open issues. Finally, Section X concludes the paper. Figure 1 shows the paper structure overview, and Table 1 shows the list of abbreviations used in the rest of the paper.

II. CONCEPTS AND DEFINITIONS

This section briefly reviews some of the leading cloud computing-related paradigms with undeniable overlaps, which hinders a specific definition for each of them due to diverse applications and use cases of these paradigms. Then, we the define dynamic service placement problem and explain its challenges and research directions. Finally, we review related concepts to the service placement problem.

A. CLOUD COMPUTING-RELATED PARADIGMS

Cloud computing is a model that promotes on-demand network access to shared computing resources, which are available in large data centres [5]. These centralised data centres offer infrastructure, platforms, and software as services that users can employ according to their needs. On-demand provisioning avoids fixed assignment that causes over or under resource provisioning.

Mobile computing offers processing capabilities using mobile, portable, and resource-constrained devices (e.g., laptops, tablets and smartphones) to provide pervasive and context-aware applications [25]. These mobile devices can be integrated into cloud data centres to form MCC environments, where there is a central entity that provides richer computing resources and introduces higher latency [26]. Similarly, **cloudlet computing** deploys small data centres (i.e., cloudlets), typically one hop away from mobile devices. These devices can decide to offload their computing tasks to closer cloudlets when needed [27]. Alternatively, **Mobile Ad-hoc Cloud Computing** proposes to form highly dynamic mobile clouds from the integration of mobile devices through ad-hoc networks. These dynamic clouds support networking, storage, and computing [28].

Mobile Edge Computing / Multi-access Edge Computing provides computing and storage resources to nearby resource-constrained mobile devices (Figure 2). It provides

TABLE 1. List of acronyms.

3CG	Three sided Cyclic Game
3GPP	3rd Generation Partnership Project
5GPPP	5G Infrastructure Public Private Partnership
5G-RAN	5G Radio Access Networks
AR	Augmented Reality
ASP	Application Service Provider
BBU	BaseBand Unit
BS	Base-Station
C-RAN	Cloud Radio Access Network
D2D	Device to Device
ETSI	European Telecommunications Standards Institute
eMBB	enhanced Mobile BroadBand
eNB	evolved Node B
EPC	Evolved Packet Core
HetNet	Heterogeneous Network
ICN	Information-Centric Networking
IIoT	Industrial Internet of Things
ILP	Integer Linear Programming
IoT	Internet of Things
MANET	Mobile Ad-hoc Networks
MEC ISG	Mobile Edge Computing Industry Specification Group
mMTC	massive Machine Type Communications
MCC	Mobile Cloud Computing
MEC	Multi-access Edge Computing
MILP	Mixed Integer Linear Programming
MIMO	Multiple Input Multiple Output
MNO	Mobile Network Operators
NDN	Named Data Networking
NFV	Network Function virtualisation
NGMN	Next Generation Mobile Networks Alliance
QoS	Quality of Service
QoE	Quality of Experience
RAN	Radio Access Network
SCeNB	Small Cell eNodeB
SFC	Service Function Chaining
SBS	Small Base-Stations
SDN	Software-Defined Network
SLA	Service Level Agreement
SOA	Service-oriented architecture
URLLC	Ultra-Reliable Low-Latency Communication
V2X	Vehicle-to-everything
VM	Virtual Machine
VNF	Virtual Network Functions
VR	Virtual Reality

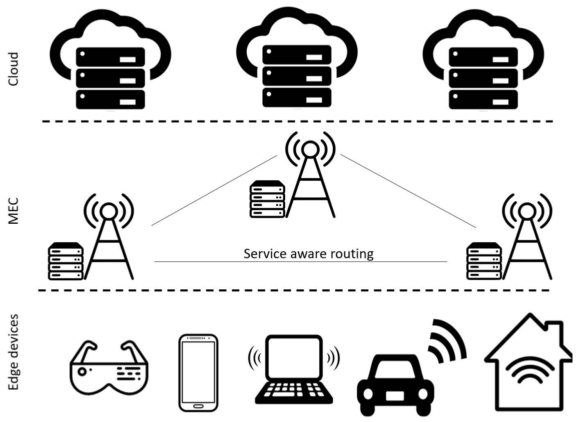

FIGURE 2. Three layers of MEC architecture.

on-demand elastic access to a shared pool of reconfigurable computing resources and interactions to nearby mobile users with minimal management effort and service provider intervention in wireless access networks [29]. This paradigm aims to enhance QoS by offering ultra-low latency environments, which reduce user-cloud data exchange and service access latency. Another characteristic of MEC is the seamless integration of multiple application service providers and vendors toward mobile subscribers, making this computing paradigm an essential component in the 5G architecture [30]. The utilisation of MEC servers distributed across network areas differentiates MEC from cloud computing that relies on centralised data centres [29]. It is also different from the mobile cloud computing paradigm that combines cloud computing, mobile computing, and wireless networks and relies on cloud infrastructure to manage ample storage and processing of mobile devices [29].

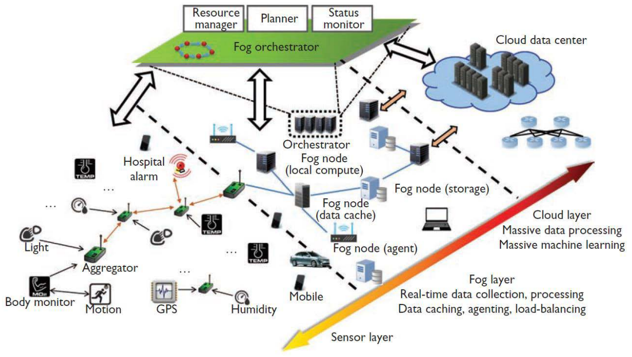

FIGURE 3. An overview of Fog computing environment [33].

Fog computing bridges the gap between centralised cloud and end devices (e.g., IoT devices) by enabling computing, storage, networking, and data management from cloud data centres for network nodes within the vicinity of IoT devices [8], [31]. The geographic distribution of such capabilities minimises an application's latency and power consumption while improving its flexibility and security compared to the traditional cloud. IEEE Standards Association adapts Fog computing reference architecture provided by *OpenFog Consortium* through IEEE 1934 [32]. Figure 3 illustrates an overview of a Fog computing environment.

Edge Computing enhances the management, storage, and processing power of connected devices (i.e., static and mobile) through small data centres closer to end-users [34]. The main difference between edge and Fog computing is that edge computing limits the computing at the local network where devices are deployed. Fog computing is hierarchical and provides computing, storage, networking, and data management anywhere from cloud data centres to user devices [35]. MEC extends mobile computing through edge computing. It provides processing and storage resources near low energy and low resource mobile devices [9]. **Mist Computing** [36] and **Cloud of Things** [37] are similar paradigms that propose dispersed computing at the extreme edge, where static and mobile IoT devices act as thin servers and thin clients in fully distributed architectures. The main difference between mist computing and the cloud of things is that IoT devices form a virtualised cloud infrastructure in the latter.

B. DYNAMIC SERVICE PLACEMENT PROBLEM

In a MEC environment, user devices are connected to MEC enabled base stations using different access technologies and

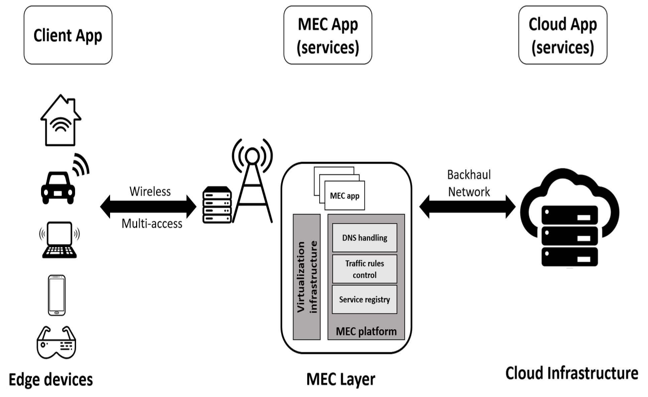

FIGURE 4. Application services in MEC paradigm.

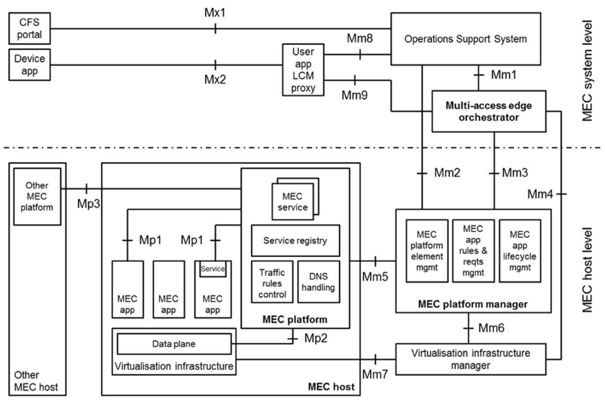

FIGURE 5. Multi-access Edge Computing framework [38].

request application services. These base stations (e.g., macro cell (eNB), small-cell (SCeNB), or femtocell (HeNB)) are equipped with edge servers (MEC servers) that host application services. The edge servers have limited computational resources compared to a cloud data centre; thus, they can only host a limited number of application services. Unfulfilled user requests are redirected to a cloud data centre via a backhaul network (Figure 4).

Figure 5 shows the main functional elements of MEC reference architecture designed by ETSI [38]. The MEC design comprises a host level and a system level. The host-level includes the virtualisation infrastructure manager, MEC platform manager, and MEC host entities. The virtualisation infrastructure manager is responsible for the lifecycle of virtual resources as well as failures. The MEC platform manager manages the MEC host and its running MEC applications. The MEC host is comprised of MEC applications, MEC platform, and visualisation infrastructure. MEC platform supports MEC applications using service registry, traffic rules control, and DNS handling, while virtualisation infrastructure provides the required computing, storage, and network resources. At the MEC system level, MEC Orchestrator maintains a global view of the MEC system, including services, applications, and resources. It also manages (instantiate, relocate and terminate) MEC applications. The operations support system provides the required functionalities for MEC service providers to communicate with devices and third party customers.

Initially, the application services are deployed in cloud infrastructure. However, they can be redeployed or migrate to the edge servers at the user proximity for several reasons, including stringent QoS requirements, considerable round trip time to the cloud, and reduced backhaul network traffic [39]. The resource limitations of edge servers and BSs hinder the deployment of all the services in an edge server. The BSs have limited bandwidth to serve the mobile users, and the edge servers have to allocate their valuable processing, memory, and storage resources in an optimum way. Some of these resources can be shared among the requests of a single service, and some of them are not shareable and require different allocation models. Another issue is the heterogeneity of edge servers in terms of their resources and their OS platforms. Finally, the coverage areas of BSs can overlap, which can be advantageous for mobile users located in the overlapped area to communicate with the one that provides a better link quality, but this introduces further issues for a controlling entity to manage the connection.

In this environment, dynamic service placement refers to the process of adaptive configuration of these services in edge servers. The main goals of dynamic service placement are to improve QoS, optimise resource utilisation, maximise throughput, maximise revenue, or combine these objects in the form of utility maximisation.

In a typical scenario, a user device requests an application service from its connected base station. The MEC enabled base station can fulfil the request providing that the requested service is cached in the MEC host component of the edge server. Otherwise, the edge server has to decide whether to forward the request to another MEC enabled base station or forward the request to a cloud data centre. The edge server decides which service to cache according to the received service requests and the characteristics of the services considering its limited computing resources. We can split the dynamic service placement problem into three sub-problems: service caching, resource allocation, and request forwarding.

1) SERVICE CACHING

Edge servers can cache several application services to reduce application latency [40] and backhaul traffic. The caching includes a service instance that is generally based on virtualisation technologies. In [41], service caching is defined as pre-storing required programs, libraries, and data to facilitate an edge server to run the computational task requested by a user device. The following questions need to be answered for the service caching sub-problem.

- Which service has to be cached? Service caching has costs and benefits. The caching costs of a service instance include rental costs of the edge server resources and the amount of bandwidth used to download a service instance from a cloud data centre. The size of a service instance depends on the utilised virtualisation technology (e.g., VM, container, function). It also depends on the size of its executable part, required libraries, and data stores. The main benefits of caching a service instance are reducing user-perceived latency, reducing backhaul traffic, and increasing revenue which implies that QoS requirements, the size of input/output data of service, and service revenue play an influential role alongside the service demand. To cache high demand services helps compensate for the caching cost, but it is challenging since demand patterns of these services are usually not known/non-stationary and changes both spatially and temporally.

- Where to place a service instance? The demand patterns of services change based on location and time. Some recent work [42] use ideas such as to cache services in edge servers based on geographical locations (location-based services), or to cache services based on users' interest, assuming that users in the same place are likely to request similar services in the near future [42].
- How many service instances are required to meet the optimum operational costs? On the one hand, edge servers have limited computing resources, and on the other hand, service instances may face failure. The failure of those service instances with strict hard real-time QoS guarantees (e.g., cooperative driving) is a severe problem investigated in [43], [44].
- When the current configuration of services has to be reconsidered? The question targets the timing of service caching decisions. One approach suggests that caching decisions to be taken at certain medium-term intervals. An alternative is an event-based caching decision, e.g., receiving a certain number of service requests.

2) RESOURCE ALLOCATION

The edge servers have to allocate three groups of resources to host and execute the user-requested services. The first group is the computational resources, including memory, disk storage, and CPU. Some of these resources can be shared among service requests. For example, once a service instance is placed in the disk storage of an edge server, all the requests for this service can use it. But, the CPU resource (e.g., a CPU core) cannot be used simultaneously by multiple requests since each request is executed separately. The second group is networking resources such as bandwidth and wireless communication channels. Finally, the last group is the budget/monetary resource for renting an edge server.

In the lack of required computational resources to host new services, the edge server has to decide whether to remove a previously hosted service instance to host a new one or not. Several criteria are proposed in the literature, including the least recently used and the least popular service.

3) SERVICE REQUEST FORWARDING

Request forwarding deals with a case in which an edge server does not host the requested service and decides whether to route the request to another base station or the cloud. The decision has to consider computation and bandwidth capacities [45] to orchestrate the request workload and balance other edge servers' load. Besides, edge servers can use the information about the cached services in their adjacency.

C. DYNAMIC SERVICE PLACEMENT CHALLENGES IN MEC

1) MOBILITY AND SERVICE DEMAND PATTERN

The user mobility and time-variant population of users nearby a MEC enabled base station make estimations of the future volume of service demands complicated. Although modelling of human mobility pattern is a topic of research for many years, erratic human mobility hinders accurate predictions of user locations.

Understanding the patterns of service demands plays a decisive role in the performance of dynamic service placement solutions, which is challenging due to user mobility, location-based user interests in particular services (i.e., location-based services), and time-based interests according to service usage (e.g., rush hour traffic routing services). Many researchers assume service demands follow a stochastic process, but occasionally large crowds may gather in a particular area during sports matches, concerts, or protests, leading to bursty user demands [46].

2) RESOURCE CONSTRAINTS

Edge servers have limited computational resources compared to a cloud data centre, making the selection of candidate services placed in an edge server challenging. Additionally, application providers have a limited budget to rent edge server resources; therefore, the revenue of each service has to be considered in the placement decision.

3) SERVICE HETEROGENEITY

Applications are heterogeneous as they have different resource demands and priority levels, which must be considered to enhance users' experience and support critical applications. Placement approaches must conciliate both sides (i.e., providers and applications) to provide services in a way that satisfies both. Application services with different QoS requirements and characteristics introduce new challenges to the dynamic service placement problem. To name a few, some applications are delay-sensitive and may require ultra-reliable low-latency communications (URLLC). Some application services are computation intensive. Some others require large input/output data sizes. Besides, service instances sizes are different. These characteristics impose challenges and complicate any candidate placement solution. Finally, application services have different inter-dependencies that cannot be ignored in placement decisions.

4) SCALABILITY
We identify two main scalability dimensions that service placement mechanisms must address. The first dimension is the number of MEC services. In the case of a large number of application services, the placement approaches must explore a large solution space and analyse a significant number of demand patterns to find the optimal solution. The second dimension is the number of edge servers which is influential in centralised methods. Although the number of edge servers in an urban area is limited, the computation complexity of the optimisation techniques (e.g., graph colouring) in placement methods may not scale to this limit.

5) SECURITY/PRIVACY/TRUST
A MEC environment faces various security and privacy issues studied in [30], [47]. Inherently, these issues raise challenges for dynamic service placement methods. The placement of multiple application services on an edge server, storing users' location/service usage, and third party ownership of edge servers are some sample cases that can potentially introduce security and privacy issues.

6) MULTIPLE ADMINISTRATIVE DOMAINS
Aside from mobile users, several stakeholders are involved in a MEC environment, such as network operators (mobile operators, core network/infrastructure operators, edge infrastructure providers), application service providers, third party edge operators, and cloud operators. Additionally, different administrative domains can manage each part of this environment. For instance, different application service providers manage edge servers. The conflict of interest between stakeholders brings up collaboration, cooperation, or competition challenges.

D. SIMILAR AND RELATED CONCEPTS
Different decentralised cloud computing paradigms share several preliminary concepts. These concepts may overlap the service placement problem due to the paradigms, application scenarios, system architecture, and networking models.

1) COMPUTATION OFFLOADING
User devices are resource-constrained, indicating that they have to offload part of their resource-intensive applications (e.g., VR, augmented reality (AR)) to other resource-rich nodes to save energy and time. This is one of the MCC paradigm's motivations that enable cloud computing for mobile users [18]. However, MCC imposes various limitations, such as additional load on the network (radio and backhaul) and high latency. Offloading the computation in an edge computing environment is introduced to address these limitations, consisting of three components. The first component, known as application/task partitioning, is responsible for deciding about one of three approaches: local execution, partial offloading, or full offloading according to various performance metrics. While this decision is challenging in a dynamic and complex network of nodes, the partitioning approach (either automatic program analysis or programmer-specified partitioning) also introduces further complications [20]. The second component deals with the task allocation, comprising of three activities. The first is to discover resource-rich nodes (e.g., edge servers) that can be used for offloading. The second is task scheduling, in which the partitioned tasks are assigned to the discovered nodes to provide optimum performance. Finally, the last activity is task placement. The last component manages different types of resources. Other classifications of these activities are also introduced in the literature [17] that answer questions such as: Can the task be offloaded? When to offload the task? Where to offload? What offloading policy will be adopted?

The main goal of computation offloading is to improve applications' performance and reduce user devices' energy consumption [17], [27], [48]. This goal reveals that computational offloading is generally from user devices to resource-rich nodes [49]. It also indicates that a user device may execute the application without the computation offloading despite the excessive consumption of resources. Compared to service placement, the service components are not necessarily offloaded from user devices. These components can be offloaded from the cloud to edge servers or any other Fog layer nodes too. Besides, user devices cannot consistently execute all the service components solely. Despite these differences, service placement and computational offloading are classified as optimisation problems, and they share similar components such as task offloading and resource allocation.

2) TASK SCHEDULING
In a Fog computing environment, Alizadeh *et al.* [50] define a job as a set of tasks where each task includes entering the information into the system, processing the data, getting access to some of the required software/infrastructures, and possibly storing data in a resource-rich node while the output returns to the user. The objective of a task scheduler is the optimum assignment of tasks to resources while several influential parameters such as task deadline, waiting time, input time, the cost for performing a job, and the energy consumption of a resource for running a task take into account [50]. In another definition [51], IoT (edge) services can be decomposed into a set of tasks, several Fog nodes can process each task, and the task scheduling problem in Fog computing discovers an optimal assignment of these tasks to Fog nodes.

In an edge computing environment, one of the differences between task scheduling and service placement is that the emphasis of the task scheduling problem is on the task requests by providing an optimal assignment of the requests to the Fog nodes that have been previously enabled to process these task requests. In contrast, the service placement problem concentrates on deploying particular services in specific nodes (e.g., edge server or Fog) that empower the nodes to process the associated service requests.

3) SERVICE MIGRATION
Service migration is considered essential to maintaining the QoS for a user that moves throughout the network [52] by moving the service to a nearby node and running the service in the user' proximity. Service migration plays an essential role in user experience for seamless use of continuous services, such as streaming services and stateful services, as the users roam around the network. While some commonalities can be found between service migration and service placement, they differ from various points. The service migration can be triggered by the mobility of a single, whereas, in dynamic service placement, the significant changes in service demand triggers the reconfiguration of services across edge servers. Additionally, dynamic service placement methods consider the service caching (copy/replication), whereas, in the service migration, the services move to another node.

4) VM MANAGEMENT
Virtualisation technologies are widely used in edge computing. These technologies (e.g., VMs or containers) provide valuable features such as platform independence, resource abstraction, and isolation [53]. Tao *et al.* [53] introduce placement and scheduling of VMs as one of the VM management aspects and is modelled as an optimisation problem to optimise instantiating expenditure, response time, energy consumption, and service providers' revenue.

5) RESOURCE MANAGEMENT
In an edge computing environment, the resource-constrained nodes are geographically distributed across the network. These nodes are designed to perform different tasks with various workloads. These complexities require efficient resource management [16] that includes several modules such as resource discovery, resource provisioning [54], resource scheduling, resource allocation [55], and admission control. These fundamental modules are further used to solve problems such as service placement.

6) CONTENT CACHING
Content caching refers to distributing content (e.g., videos) in multiple servers [48], [56]. Similarly, website caching stores web pages' information reduces web browsing time [57]. The edge computing environment can benefit from content caching and, in a more general category, ICN to reduce access latency, backhaul traffic and offer better QoE. The MEC nodes can be used as content caching nodes to store popular contents frequently requested by users [24]. Content caching and service caching have several commonalities. However, it is worth noting that content caching strategies can differ from service caching ones because of the stored entity. Service caching stores application services (i.e., encapsulated pieces of code) that perform atomic functionalities. These functionalities require various resource types (e.g., CPU and memory) that have to be considered in the service caching strategy since they require different resource usage and allocation models from content caching, where storage is the central concern. For instance, if content is cached on a node, the volume of requests for this content does not affect the amount of allocated storage resource, while if a service is cached on a node, the volume of requests for this service has to be considered in the caching strategy.

III. RELATED SURVEYS
Several surveys have captured various aspects of the service placement problem in different distributed computing paradigms. We categorise them into three groups. The first group includes the research that addresses the service placement problem in other computing paradigms than MEC. In the second group, we review the works based on related/similar problems in the MEC environment. Finally, the last group comprises the research in the MEC environment that covers a broader scope. Our work is the first research that has fully dedicated to dynamic service placement problem in the MEC environment to the best of our knowledge.

A. SERVICE PLACEMENT SURVEYS
Wittenburg *et al.* [58] define the service placement problem as finding the most suitable network node for service hosting. A candidate service placement solution has to answer the following four questions:

- Where to place a service instance?
- How many service instances should be deployed?
- When to adapt a service configuration?
- How to transfer the service state to another node?

The authors explore ten representative approaches for service placement in mobile ad-hoc networks (MANET) to deal with service placement requirements by considering these objectives. The authors conclude that the existing state-of-the-art does not include mechanisms to address all the characteristics of the service placement problem.

Ali *et al.* [59] envision service placement as one of the essential features of future mobile communication networks and survey service placement methods for these networks. They consider the service placement task a *k*-median problem and a facility location problem and qualitatively assess literature that solves these problems. The analysed works are organised based on their strategies to deal with: centralised, distributed, and hybrid placement. The centralised algorithms make two assumptions: that the network topology is known and that the cost of each connection is assigned. The authors focus on three algorithms: random, greedy, and K-min. They conclude that such algorithms rely on complete knowledge about the topology and service client distribution, with potentially limited scalability in dynamic networks because of the large number of messages that need to be exchanged between service participants. Distributed algorithms were introduced to address the limitations introduced by centralisation. These algorithms incrementally adjust the placement of services

TABLE 2. Review of service placement surveys.

Work	Year	Research	Environment			Covered layers			Evaluation	Taxonomies		Open
		directions	MEC	Fog	MANET	Networking	Middleware	Application	methods	Objectives	Optimisation	challenges
[58]	2008				✓		✓					✓
[59]	2010				✓		✓					✓
[60]	2020			✓			✓				✓	
[61]	2020			✓			✓			✓	✓	✓
[62]	2020			✓			✓		✓		✓	✓
Our work	2022	✓	✓			✓	✓	✓	✓	✓	✓	✓

to optimise for particular objectives using local knowledge. An identified problem with this type of algorithm is the reduced reactiveness which may lead to sub-optimal solutions. Hybrid mechanisms combine the two types of algorithms to leverage the advantages of both approaches. The authors identify that the existing proposals do not address the single point of failure problem introduced by the centralised approaches.

Raghavendra *et al.* [60] present a short survey of optimisation techniques for application module placement in a Fog environment. They classify these optimisation techniques to exact methods, heuristic methods, hybrid methods, and hyper-heuristic techniques. Then, they review the Fog service placement methods based on different control parameters, including makespan time, cost, energy, service deadline, fault tolerance, scalability and network usage.

Brogi *et al.* [61] review the works that address optimally placing application components based on different (sometimes orthogonal) constraints in a Fog computing environment. These works are analysed based on their applied algorithms, the size of the test use cases, and the availability of the prototypes/experiments. They also analyse the works based on their modelling perspective, including functional and non-functional constraints and optimisation metrics. However, the authors exclude research in task offloading and do not study different networking technologies to support application placement in a Fog environment. The authors identify a few distributed methods for the Fog application placement problem and propose that devising decentralised/distributed methods may offer better scalability and resilience in highly dynamic Fog infrastructures. Besides, they suggest designing a set of benchmark examples to compare and contrast different solutions, providing better insight into performance improvements or degradation. Finally, they raise concerns over ignoring the security considerations in most of their surveyed methods.

Salath *et al.* [62] explore different service placement methodologies and strategies in a Fog computing environment. The authors introduce an infrastructural model, application model, and deployment pattern of Fog computing. They define service placement as a mapping pattern by which application components and links are mapped onto

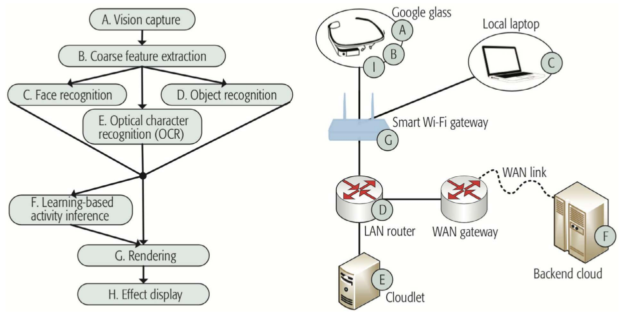

FIGURE 6. Mapping application components to an infrastructure graph [62].

an infrastructure graph, as depicted in Figure 6. The defined service placement can be static in which the deployment decision is made in compile-time, or it can be dynamic where the decision is based on runtime information. The authors provide a taxonomy of solutions based on control plan design, placement characteristics (online vs offline), system dynamicity, and mobility support. The authors conclude with open research challenges as future directions. To our knowledge, this is the most comprehensive survey in the Fog service placement problem. To compare this work with ours, both works introduce a taxonomy of design objectives, a taxonomy of solutions, and research directions. However, the difference is in the computing paradigm, where they focus on Fog computing while our work focuses on the MEC environment. Service placement problem in Fog computing environment considers deploying Fog-based applications by mapping application components to network nodes including multiple Fog nodes and the cloud infrastructure. In contrast, dynamic service placement in a MEC environment concentrates on reconfiguring a mobile cloud service among edge servers, including service caching and service offloading (from cloud/user devices to edge servers). Another enhancement of our work is investigating different service instance packaging technologies and highlighting their key characteristics that influence the dynamic service placement methods. We also study different network technologies and their support for dynamic service placement.

Table 2 highlights the contributions of different service placement surveys.

B. SURVEYS ON RELATED PROBLEMS IN EDGE COMPUTING

1) COMPUTATION OFFLOADING

Computation offloading is a process that helps mobile devices save energy and time by executing some of their energy/computation-intensive tasks on resource-rich nodes such as edge servers or the cloud.

Mach *et al.* [18] focus on computation offloading by reviewing the works from three different viewpoints. The first view concentrates on whether computation offloading is profitable for end-users (e.g., energy, time). The answer can be a local execution, partial offloading, or full offloading. In the second view, the authors study the works based on the efficiency in allocation computing resources. Finally, the last view studies service continuity of the offloaded applications as the end-users roam throughout the network.

Aazam *et al.* [63] present a taxonomy of task offloading schemes that have been proposed for Fog, cloud computing, and IoT domains. They also include a discussion on middleware technologies that have been designed to enable offloading in cloud-IoT cases. Factors that are considered when making offloading decisions are discussed. Typical scenarios where offloading may be necessary are included. The authors conclude with future research challenges that need to be addressed to improve task offloading performance, efficiency and reliability in Fog computing.

Wang *et al.* [19] introduce a new characterising model for the computational offloading process. In their five-dimensional model, they study offloading destination (single vs multiple servers), offloading balance (offline vs online), mobility of devices, offloading partition (static, dynamic, or hybrid), and partition granularity (application, task, or method level).

Lin *et al.* [20] study the challenges for computation offloading in edge computing from application partitioning, task allocation, and task execution perspectives. Then, they investigate different application scenarios for computation offloading, including video analytics, smart *things*, intelligent vehicle applications, and cloud gaming. Finally, they propose a programming model, offloading as a service, and security/privacy as future directions.

Jiang *et al.* [17] present a literature review of computation offloading and resource allocation approaches for edge computing, supported by a presentation of several case studies. The authors consider energy consumption minimisation, QoS guarantees, and achieved QoE enhancements. In particular, there is a focus on resource scheduling approaches, overhead and system performance overhead, with limited analysis of dynamic service placement.

Lin *et al.* [64] focus on modelling for computation offloading by studying the basic models, including the channel, communication, and energy harvesting models. They analysed different modelling methods such as (non-)convex optimisation, Markov decision process, game theory, Lyapunov optimisation, or machine learning. The analysis reveals that most computational offloading models use the state-action model. They argue that reinforcement learning-based schemes that employ artificial intelligence have some advantages over others due to the problem features such as high complexity, large scale, and unpredictability.

Shakarami *et al.* [65] review the machine learning-based computational offloading methods in MEC environments and present their advantages and limitations. They classify these methods into reinforcement learning, supervised learning, and unsupervised learning and investigate their covered offloading metrics, including energy, latency, Quality of Service, Quality of Experience, response time, and cost.

Wang *et al.* [49] study the task offloading problem in edge-cloud computing from various views, including which tasks are offloading from user devices, which edge or cloud is an offloading task assigned to, and which server each offloading task performs on in what sequence. They introduce a taxonomy of methods comprising task types, offloading schemes, objectives, device mobility, and multi-hop cooperation.

2) TASK SCHEDULING IN EDGE COMPUTING

A task scheduling method aims to discover the optimum assignment of application components to network nodes for execution. Hosseinioun *et al.* [51] survey the current research on task scheduling in edge (Fog) computing environment and study various aspects, including architectures, algorithms, and their related advantages and limitations. The authors introduce a taxonomy of task scheduling methods and raise open issues and challenges such as security, privacy, trust, and fault tolerance. In a similar survey, Sindhu *et al.* [66] study different task scheduling techniques and approaches in Fog computing. Alizadeh *et al.* [50] present a systematic literature review of scheduling tasks in a Fog environment. They review the task scheduling algorithms and identify several metrics that each algorithm tries to improve. Moreover, they propose a taxonomy based on the architectures and the algorithms of task scheduling research in the literature. Finally, they introduce open perspectives and future challenges, including resource management, energy management, security, and QoS. Islam *et al.* [67] review context-aware scheduling in Fog computing environments. The authors define scheduling as selecting appropriate resources for the tasks, meeting the minimum completion time, and improving the throughput to satisfy the users' QoS requirements. Task scheduling, which allocates appropriate resources to the tasks submitted according to scheduling goals, is challenging due to resource heterogeneity, dynamic nature, resource limitations, and unpredictability of the Fog environment. The context information of the entities involved in Fog computing can be used to address these challenges. The authors analyse different levels of contextual information and introduce a taxonomy of context-aware parameters related to entities, including user, application, environmental, network, and device.

3) SERVICE MIGRATION

The mobility of users in a MEC environment and the continuity of service usage require that these services migrate to an edge server in the user proximity. Wang *et al.* [39] compare two similar concepts (live migration for data centres and handover in cellular networks) with service migration. The first one deals with memory migration of virtual machine instances that help improve the utilisation of resources, load balancing of processing nodes, and tolerate the faults in virtual machines in data centres. The second one deals with seamless communication handover (or hand-off) of a mobile user who moves from one cell to another. Then, the authors provide a taxonomy of various research directions of service migration, including the techniques of migrating running services and the strategies for service migration. The authors also explore three main application component hosting technologies. Finally, they conclude by highlighting the potential areas for further research. Rejiba *et al.* [52] present a taxonomy of mobility-induced service migration mechanisms available at different edge-centric computing paradigms, including cloudlets, Fog computing, cloud-based vehicular networks, and multi-access edge computing. The authors focus on the optimisation techniques employed by these mechanisms and classify design objectives. Additionally, the authors study the works concentrated on architectural design and implementation. We believe the service migration problem has an overlap with dynamic service placement. However, in our work, we consider that the dynamic service placement problem can include system dynamics generated by other than users' mobility (e.g., availability of edge devices).

4) VM MANAGEMENT IN EDGE COMPUTING

VM technologies are one of the main enablers of edge computing. Tao *et al.* [53] present an overview of VM management in edge computing, including virtualisation frameworks and techniques, serverless computing, and the security advantages and issues based on industrial and research projects. In the virtualisation techniques, the authors review various research projects such as VM (system virtualisation), Linux containers (kernel virtualisation), service mobility (geographical virtualisation), and VM-based SDN. The authors also introduce several design objectives for VM placement and scheduling problems in an edge computing environment.

5) RESOURCE MANAGEMENT IN EDGE COMPUTING

Teng *et al.* [55] study the resource allocation problem in ultra-dense networks, including HetNets, C-RANs, massive/large scale multiple input multiple output (MIMO) networks, device to device (D2D) networks, and massive IoT environments. They provide a taxonomy of resource allocation methods and introduce emerging technologies. The authors conclude by presenting challenging issues such as high computational complexity, significant overhead and feedback, spatio-temporal dynamics, mobility management, interference management, channel models, and QoS/QoE requirements. Duc *et al.* [54] investigate resource provisioning in joint cloud-edge environments and present a survey of the technologies and methodologies used to enable applications in such environments. The authors break down the resource provisioning problem into three main components: workload prediction and characterisation, component placement and system consolidation, and application elasticity and remediation. Hong *et al.* [16] present a survey of work on resource management. The focus is on documenting the architectures and the underlying algorithms used by these approaches. There is also a perspective on the evolution of the techniques, with papers published in 1991 included. While some of those techniques are outdated, the authors produce a histogram of these papers to show the most attention areas.

6) CACHING IN MEC

Parvez *et al.* [68] survey the emerging technologies to achieve low latency communications towards 5G and introduce caching as one of these solution domains that can mitigate the insufficient capacity of backhaul links as well as long delays due to an excessive number of user requests. The authors present a detailed overview of content caching for cellular networks as well as the fundamental limits. Yao *et al.* [24] review the research on content the caching problem and indicate that edge servers provide computing and storage resources that can be used in content caching. They investigate the content caching works according to a four-step process: content request, exploration, delivery, and update. Piao *et al.* [69] review the edge cache in RANs by analysing the deployment location of edge caches, content placement strategy, and coded caching. For deployment locations, they classify the survey methods into three groups of edge cache at BSs, edge cache at mobile devices, and joint edge cache at both BSs and mobile devices. The content placement strategy is surveyed based on reactive and proactive strategies. Finally, they study coded edge caching that, unlike traditional caching in which the entire content must be cached at an individual location, the files are split into several segments and save part of them in user devices. In the delivery phase of coded edge caching, each user requests a file in the database, and all the requests are coded together and transmitted to all users. The authors highlight the performance gain of edge caching and introduce future challenges.

The main difference between our work and the above is that they do not cover service caching and, in a more general view, service placement in their works. It is worth recalling that although there are some common aspects between content caching problem and service caching problem, we point out the differences in the *excluded subjects* part of Section I-E that differentiate these problems.

7) EDGE COMPUTING SECURITY

Edge devices are vulnerable to attacks due to various reasons, including weak computation power, attack unawareness,

TABLE 3. Surveys on related problems in an edge computing environment.

Work	Year	Computing paradigm	Computation offloading	Task scheduling	Service migration	VM management	Resource management	Content caching	Security	Service placement
[18]	2017	MEC	✓							
[63]	2018	Fog computing	✓							
[19]	2019	Edge computing	✓							
[20]	2019	Edge computing	✓							
[17]	2019	Edge computing	✓							
[64]	2020	Edge computing	✓							
[65]	2020	MEC	✓							
[49]	2020	Edge-Cloud Computing	✓							
[51]	2019	Fog computing		✓						
[50]	2020	Fog computing		✓						
[66]	2020	Fog computing		✓						
[67]	2021	Fog computing		✓						
[39]	2018	MEC			✓					
[52]	2019	Edge computing			✓					
[53]	2019	Edge computing				✓				
[55]	2018	Ultra-dense network					✓			
[54]	2019	Edge-Cloud computing					✓			
[16]	2019	Edge computing					✓			
[68]	2018	Edge computing						✓		
[24]	2019	Edge computing						✓		
[69]	2019	Internet of things						✓		
[70]	2019	Edge computing							✓	
Our work	2022	MEC								✓

OS/protocol heterogeneity, and coarse-grained access control [70]. Xiao *et al.* [70] study four basic types of attacks: distributed denial of service (DDoS) attacks, side-channel attacks, malware injection attacks, and authentication and authorisation attacks. These attacks present 82% of edge computing attacks. The authors also review the corresponding defence mechanism for these attack types.

See Table 3 for a list of other related surveys.

C. SURVEYS WITH A WIDER SCOPE

The last group is the dedicated MEC environment surveys that cover a more extensive scope by studying a wide range of problems.

Liu *et al.* [29] present a survey of MEC systems that introduces MEC applications and characteristics and service models. They also explore the influencing factors of MEC system design and survey different MEC architectures, including MEC server and network architecture.

Wang *et al.* [71] provide an overview of MEC architecture and highlight the differences with other computing paradigms such as mobile cloud computing and Fog computing. They review research efforts on computation offloading, edge-core cooperation, and the combination with 5G. They also survey the works on content caching, covering various aspects, including content popularity, caching policies, scheduling, and mobility management. Additionally, several MEC-based categories of applications (e.g., dynamic content delivery, AR/VR, video streaming) are introduced.

Mao *et al.* [40] summarise the existing communication and computation models in MEC to overview the state-of-the-art for both researchers and practitioners. They extend this summary with a review of joint radio-and-communication resource allocation mechanisms. The authors focus on the communication perspective of the resource management problem and present several research challenges and opportunities in MEC.

Taleb *et al.* [48] introduce MEC and its key enabling technologies. They review MEC frameworks and reference architecture and investigate MEC service and network orchestration deployment options. Finally, they review standardisation activities and elaborates further on open research challenges

Porambage *et al.* [72] concentrate on IoT applications and provide a comprehensive view of exploiting MEC technology for this category of applications. They present future directions in scalability, communication, computational offloading, resource allocation, mobility management, and security. Moreover, they review MEC integrating technologies, including NFV, SDN, ICN, and network slicing.

Abbas *et al.* [30] review different characteristics of MEC and survey the current and emerging application scenarios in AR, healthcare, video analysis, connected vehicles, etc. The authors also review the conducted research on MEC infrastructures, such as deployment scenarios and testbeds. They also study the security issues of MEC, including security concerns, security mechanisms, and privacy issues.

Hassan *et al.* [73] focus on edge computing in 5G by establishing a taxonomy that addresses different viewpoints such as objectives, computational platforms, attributes, use of 5G functions, the role of edge computing in 5G, and performance measures. The authors introduce open issues that need to be addressed, such as service enhancement (QoE), standardisation, heterogeneity, and security vulnerabilities.

Zhao *et al.* [74] concentrate on infrastructures and applications of edge computing. First, they study edge computing issues from infrastructure, control and management, resource virtualisation, and system architecture viewpoints. They also investigate service provisioning and performance optimisation challenges of providing high QoS. Finally, they review edge computing applications and propose a taxonomy of them.

Filali *et al.* [75] review the integration of MEC into the architecture of current mobile networks and the transition mechanism to migrate into a standard 5G network architecture. They study the role of SDN, NFV, SFC and network slicing as complementary technologies to MEC. They also investigate the methods for MEC resource optimisation and propose an architectural framework for a MEC-NFV environment.

Pham *et al.* [76] provide an overview of MEC technology and identify three types of use cases: consumer-oriented

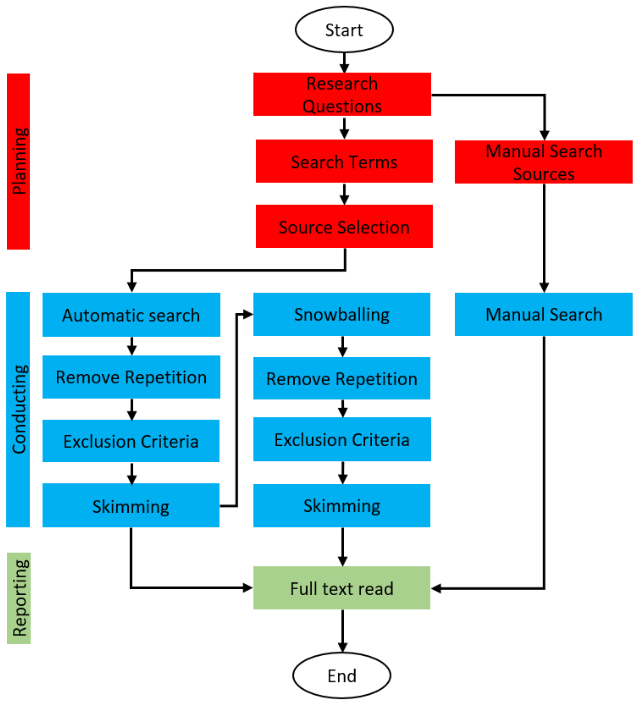

FIGURE 7. An overview of the research method.

services, operator and third-party services, and network performance and QoE improvements. They review the integration of MEC with the new technologies deployed in 5G and beyond. Then, they summarise open source activities, testbeds, and experimental evaluations.

Liu *et al.* [77] analyse MEC from 5G and IoT viewpoints and study various technologies including cloud computing, SDN, NFV, ICN, VM/containers, smart devices, networking slicing, and computation offloading that facilitate MEC integration into 5G and IoT. They also elaborate several MEC enabled applications, including intelligent transportation systems, AR/VR/MR, smart grids, factory automation, serious gaming, healthcare, smart home, smart city, smart farming, and smart retail.

IV. RESEARCH METHOD

Systematic literature review (SLR) is a well-established methodology to identify, select, and report all the studies on a specific topic. We adopt published guidelines and organise three research stages: *Planning*, *Conducting*, and *Reporting* [78], [79]. Each stage includes several activities described as follows and illustrated in Figure 7.

A. PLANNING STAGE

In the first stage, we define the research questions, identify the search terms, and select scientific databases.

1) RESEARCH QUESTIONS

Considering dynamic service placement as a middleware component that uses network layer capabilities and serves the application layer, we formulated five research questions covering different aspects, where each question sheds light from a different perspective.

- **RQ1**: What is the effect of networking technologies on dynamic service placement solutions? (Section V).
- **RQ2**: What are the current research directions in dynamic service placement? (Section VI-B)
- **RQ3**: What is the taxonomy of dynamic service placement methods? (Section VI-C and Section VI-D)
- **RQ4**: Which category of applications can benefit from dynamic service placement? (Section VII)
- **RQ5**: What are the current research validation/evaluation methods? (Section VIII)

2) SEARCH TERMS

We consider two sets of terms. The first targets distributed computing environments for dynamic service placement. The second refers to terns related/similar to *dynamic service placement*. We choose a wide range of terms since they are occasionally used interchangeably.

Set 1: {*Multi-access Edge Computing*, *Mobile Edge Computing*, *MEC*, *Internet of Things*, *Fog Computing*, *Edge Computing*, *Mobile Computing*, *Cloudlet*, *Mist*}

Set 2: {*Service Placement*, *Function Placement*, *Service Offloading*, *Service Caching*, *Elastic Deployment*, *Dynamic Service Deployment*, *Service Orchestration*}

3) SOURCE SELECTION

In the planning stage, we select the candidate digital libraries (scientific databases) for search sources. We choose four well-known computer science publishers, including *IEEE Electronic Library*, *Elsevier* (*Science Direct*), *SpringerLink*, and *ACM Digital Library*. Additionally, to cover other publishers, we use *Scopus* and *Engineering Village* indexers to have an inclusive search space.

4) MANUAL SEARCH SOURCES

We verify the following journals that regularly publish related surveys. The journals are used in the manual search to extract similar surveys to our work.

- ACM Computing Surveys
- Computer Communications
- Computer Networks
- Future Generation Computer Systems
- IEEE Access
- IEEE Communications Surveys & Tutorials
- IEEE Internet of Things Journal
- Journal of Network and Computer Applications

B. CONDUCTING STAGE

The conducting stage focuses on the search and selection process. We apply three search approaches to collect all the related studies, including automatic search, snowballing, and manual search. We use automatic and snowballing methods to probe studies on dynamic service placement in MEC.

TABLE 4. Selected peer-reviewed research databases.

Source	Search fields
ACM Digital Library	Title, Abstract, Keywords
Engineering Village	Subject, Title, Abstract
IEEE Electronic Library	Document title, Abstract, Index terms
Science Direct	Title, Abstract, Author-specified keywords
Scopus	Article title, Abstract, Keywords
SpringerLink	All fields

TABLE 5. Number of retrieved papers on automatic search.

Research database	Number of papers
ACM	207
Engineering Village	479
IEEE	239
Science Direct	187
Scopus	315
Springer	404
Total	**1831**

The automatic and snowballing search outputs are checked for repetition, exclusion criteria, and skimming to filter out the unrelated ones. We also apply automatic and manual methods to identify similar surveys to our literature review. Table 6 shows the number of papers in each conducting stage.

1) AUTOMATIC SEARCH

The conducting stage starts with an automatic search from the selected databases. These databases provide different searchable fields, and Table 4 depicts the ones used for each database. Using the search terms, we consider papers containing at least one term from each set. Table 5 presents the numbers of retrieved papers from each database according to the search performed in March 2021.

2) SNOWBALLING

Both forward and backwards snowballing methods are used. The input for this activity is the list of papers that passed the selection criteria with repetitions removed, exclusion criteria checked, and paper skimmed. In backwards snowballing, we check the references of all the papers in the input list. Then, we extract the papers that passed the selection criteria. In forward snowballing, we review the citations of every paper in the input list using *scholar.google.com*. The outcomes are checked against the selection criteria. The successfully-passed papers are added to the repository of selected papers.

3) MANUAL SEARCH

A manual search is performed to discover any previously published surveys on dynamic service placement or related areas. We look for any surveys or review papers that addressed our topic specifically or addressed a broader topic that includes dynamic service placement.

TABLE 6. Number of papers in each stage.

Conducting stage statistics	Number of papers
Initial automatic search	1831
after pruning duplicates	1180
after exclusion criteria and skimming	**87**
Manual search	**24**
Initial snowballing	69
after exclusion criteria, duplicates, and skimming	**25**
Total remaining papers	**87+24+25=136**

4) REMOVE REPETITIONS

We import all the retrieved papers into the *Mendeley* application and remove repetitions.

5) EXCLUSION CRITERIA

We apply the following exclusion criteria to filter out unsuitable papers:

- Papers that are written in languages other than English
- Non-citable documents
- Papers with less than three pages
- Papers without not peer-reviewed
- Papers that were a duplicate of others
- Papers not accessible in full-text

6) SKIMMING

The last selection criterion is to skim the paper to decide whether the content is related. We take into account the scope and domain introduced in Section I-E.

C. REPORTING STAGE

In the last stage, we read the full text of the papers in the finalised list and investigate the research questions.

V. NETWORK LAYER PERSPECTIVE

This section elaborates on the dynamic service placement problem from the networking perspective.

A. 5G NETWORKS AND MEC ARCHITECTURE

In recent years, the telecom world has undergone tremendous growth in developing mobile and wireless network communications technologies. With highly capable user devices (e.g., smartphones and tablets) and new interactive real-time applications in place, this is the midst of a data and application revolution. This is also reflected by the fact that 68% of the world population currently has a mobile subscription [1]. The arrival of the 5G and its diverse aims compared to previous standards opens the door to a rapid evolution of the Internet, a new generation of applications and technologies. 5G communications can be categorised into three main categories: enhanced mobile broadband (eMBB), URLLC, and massive machine-type communications (mMTC). Besides, 5G is required to prop up communications, control, computing, content and service delivery. Holographic interfaces and AR are a few applications that will benefit from the massive com-

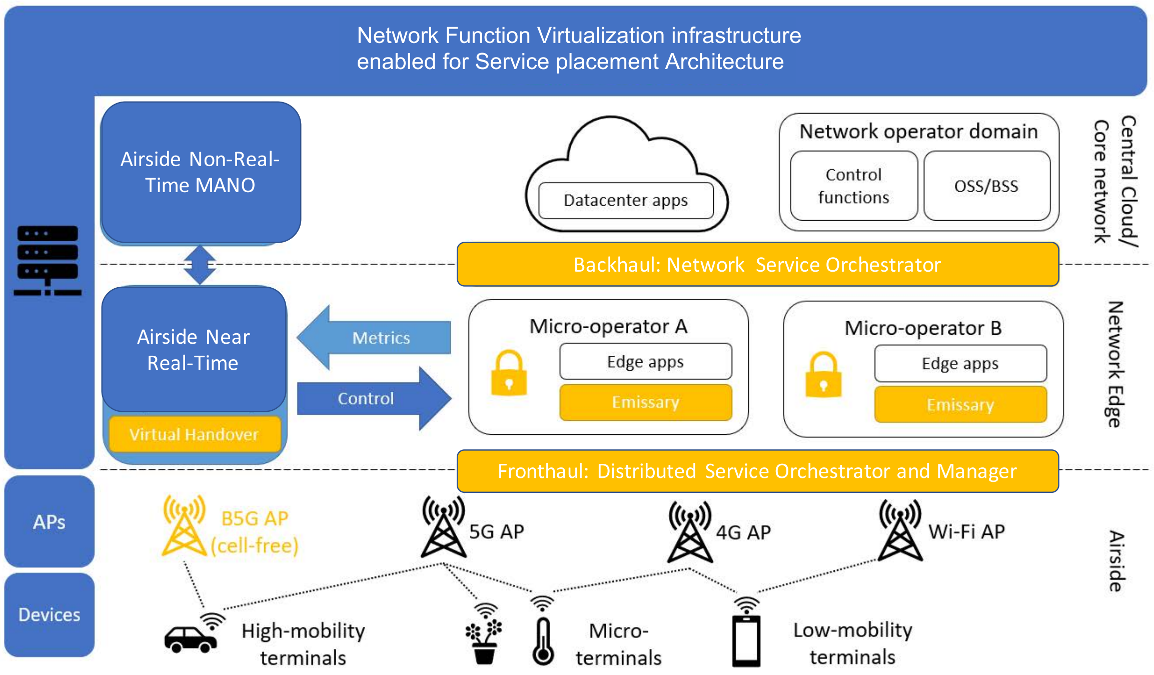

FIGURE 8. An illustration of service placement in MEC supported network architecture.

munication bit rates and ultra-low latency features expected to be provided by 5G [48]. MEC will be a crucial component in the 5G system and network architecture to support such constrained and dynamic services [80]. Despite the benefits of 5G networks, it will take some time until the existing 4G system transitions to a complete 5G system. Commercial 5G networks are initially deployed in non-standalone mode, meaning that the 5G RAN interfaces with the 4G evolved packet core (EPC) to make higher bit rates available. However, this approach hinders the deployment of new applications and services at the network edge since the standard EPC lacks proper support for edge application services instead of a native part of the 5G core.

It is essential for mobile network operators (MNOs) to reduce the time-to-market of new applications for overall revenue maximisation as well as the participation and collaboration of multiple stakeholders (e.g., mobile operators, service providers, and users). Pham *et al.* [76] group the major growth drivers in the MEC market into four major categories: technical integration, potential use cases, business transformation, and industry collaboration. The MEC applications are expanding in the new markets for various industries and sectors by supporting a wide range of use cases, including IoT, Industry 4.0, vehicle-to-everything (V2X) communication, Smart Cities, and Tactile Internet. A high-level technical integration of MEC, including characteristics, architecture and applications, is pictorially illustrated in Figure 8. Despite several opportunities and potentials, many challenges need to be studied to create an edge ecosystem where all network players (e.g., micro/mobile terminals, service deployment and placement entities, infrastructure providers, and mobile operators) can benefit from niche networks edge services. The discussion can be summarised as follows.

1) CELLULAR NETWORK AND MEC INTEGRATED DEPLOYMENT

MEC servers can be deployed at different places within the RAN depending on the underlying wireless/cellular technologies, technical and business requirements. Thus, another critical challenge is the seamless integration of MEC into the underlying network architecture and multiple wireless interfaces. The existence of MEC and enabled applications should not affect the standard specifications of the core network and end devices. The critical component of the MEC integration is the ability of MEC to interact with 5G networks in routing the traffic and receiving relevant control information [76]. Furthermore, application migration necessitates a so-called application portability requirement to prevent service providers from designing multiple versions for different MEC platforms.

2) HETNET - A COEXISTENCE OF DISTRIBUTED MEC AND CENTRALISED CLOUD

Cloud having abundant computing resources, can process big data applications and services in temporal constraints while supporting many users and devices. However, distributed MEC is highly desired since the computation at the network edge can meet the user requirement and reduce the end-to-end delay for near real-time services caused by the traffic congestion and transmission delay. An HetNet architecture, as depicted in Figure 8, can be highly beneficial to implement MEC in a hierarchical structure, i.e., Airside (i.e. user/devices/terminals/APs), network edge (i.e. computing, multi-operators, QoS/QoE metrics and requirements security), and cloud layers (i.e. cloud/core network). In this way, the MEC vendor also injects computing resources to the small-eNBs (orchestrator) so that the advantages of HetNets can be exploited for diversifying multi-radio transmissions, spreading computing demands and supporting multi-service provision and orchestration. It is important to note that distributed MEC may not have enough computing resources to process all computation requests. Complete reliance on the cloud constrains provision of latency-critical services. Therefore, it is intuitive to distribute big-data/latency-critical computations to distributed MEC servers while transferring compute-intensive and delay-tolerant non-real-time tasks to the cloud [23], [81], [82]. The coexistence of distributed MEC and centralised cloud is an essential feature for a service-efficient HetNet architecture.

3) HETNET ARCHITECTURAL COMPONENTS

The physical HetNet infrastructure in Figure 8 consists of three tiers: (i) macro-terminals/devices with no/limited computational capabilities (sensors and actuators), (ii) set of resources that possess computational power and/or storage capacity (MEC servers, service provider network orchestrators), and (iii) core cloud. Any device equipped with computational resources (e.g., CPU, memory, storage, and bandwidth) is potentially an edge node, such as routers, small servers, access points or gateways. HetNet is a promising architecture, providing better coverage, higher bandwidth, reduced latencies, and higher uplink and downlink data rate to end users [55], [76] due to the integration of higher-tier macrocells and lower-tier small cells.

4) THE NETWORK CONTROL PLANE AND USER PLANE

From the HetNet perspective, converged network coordination is a core element for adequate service management and placement strategy. Two common network user and control planes in reference to the centralised and distributed computing paradigm are here:

- *Centralised control plane*: A centralised control plane requires global information about application services, associated demands and infrastructure resources to take and disseminate deployment decisions. While the advantage of centralised placement is to find a globally optimal solution, they are vulnerable regarding the scalability and the computational complexity issue. When surveying papers, we observed that majority works (according to Tables 11, 12) consider a centralised network control plane when addressing the service placement.
- *Distributed user plane and control plane*: Unlike a centralised solution, a distributed approach considers multiple factors, such as users, authorities, service providers, and orchestrator nodes, to control the service provisioning and availability. Generally, distributed in the network, the network management elements compute placement decisions based on local resources and information. These network planes are more flexible and can be more efficient to handle the dynamic changes in network infrastructure. The distributed approach helps address the user/device scalability and locality awareness issues and allows placement services that best fit the local context. However, no guarantees are provided regarding the global optimum in such solutions. The researchers in [83], [84] consider an edge architecture composed of edge nodes for coordination among the works that apply a distributed control network plane. The works deal with the tasks that need real-time processing. The complex tasks that require more computational capabilities are forwarded to the cloud.

These network coordination approaches are relatively different in both centralised and distributed systems and have their advantages and limitations.

B. SERVICE-ENABLING NETWORK TECHNOLOGIES IN 5G MEC

Several key enabling technologies support MEC to be integrated with 5G. These technologies include SDN, NFV, ICN, and network slicing.

1) SOFTWARE DEFINED NETWORKING

The SDN facilitates the network technology to be programmable and agile [80] while segregating the data plane and the control plane. The control plane in an SDN controller has a global view of the whole network to orchestrate (control and manage) and control functionality in the network entities (i.e., controllers) to progress the concept of network softwarization, which can offer programmability capabilities to authorised tenants [48]. SDN supports the enhanced management needed on the MEC platforms. Due to the software functionalities in the SDN controllers, the controllers communicate more frequently with the data plane entities, i.e. end-devices. Therefore, SDN controllers must be located in the proximity of the end-devices to satisfy the service requirements, which maps the requirement of MEC where the infrastructure resources are deployed close to the end-devices, thereby providing low-latency services. Additionally, the SDN controller needs to handle virtualized service instances (i.e., VNFs, VMs, and containers) allocated and re-located dynamically, enabling SDN to support flexible service chaining, offering a dynamic service provisioning by connecting VNFs and MEC services [48]. Filali *et al.* [75] define five advantages of using SDN in a MEC environment, including scalability, availability, resilience, interoperability, and extensibility.

The significance of SDN integration with MEC in dynamic service placement is studied and verified by several research [85]–[87]. Yin *et al.* [85] assumed an SDN controller controls all base stations, then they introduce a cooperative framework that optimises the routes of the raw data, configures the hosted services, and arranges the data processing. Chalapathi *et al.* [86] utilise a wireless SDN network for a multi-cloudlet network to minimise latency and enhance QoS for mobile devices. Ouyang *et al.* [87] use an SDN controller to collect all service request information and determine the optimal MEC node to serve a user.

2) NETWORK FUNCTION VIRTUALISATION (NFV)

An idea of complementary technology, NFV, decouples the physical network hardware from the network functions and services by virtualisation. This is achieved by shifting the network functions from specialised networking hardware to commodity computing hardware using softwarization [88]. This approach is efficient for functions that require frequent changes over time. NFV enables a given service decomposed into a set of VNFs implemented in software and executed on servers [89]. It also provides flexibility in both the data plane and control plane by scaling up and down the allocated resources according to service demands [48]. Besides, NFV enables different ways to realise flexible hardware-agnostic network service provisioning. The NFV deployment characteristics include 1) flexible network function deployment, 2) software decoupling from hardware, and 3) dynamic control and management [90].

Several researchers address the integration of NFV into MEC architecture. Yang *et al.* [91] show that NFV can enhance MEC flexibility by deploying MEC services to several nodes capable of their resource virtualisation. Their framework simulated an NFV integrated MEC, allocating resources for time-sensitive services while satisfying latency requirements. A double tier architecture integrating NFV with MEC is proposed in [92], namely MANO+, which can enhance MEC services. Chiti *et al.* [93] propose two virtual function placement strategies to minimise both the worst application overall completion time and the number of applications exposed to an overall completion time greater than their deadlines. Bhamare *et al.* [94] introduce a virtual network function placement method to reduce the total delays and minimise the overall costs to operate C-RANs. Jia *et al.* [95] consider a metropolitan area network with distributed Cloudlets that uses VNF services. They propose a QoS-aware method that maximises the number of admitted requests and minimises their admission cost within a finite timeline. Subramanya *et al.* [96] introduce a neural-network-based method for scaling and placement of virtual network functions.

3) INFORMATION-CENTRIC NETWORKING (ICN)

Satisfying the surge in demand of the explosive big data in the emerging and dynamic services (e.g., AR/VR, autonomous and connected driving, and drones), new networking technologies such as ICN [97] are proposed and developed over the past decade. ICN uses a novel publish-subscribe system (information-centric) to differ from the traditional client-server system (host-centric). ICN proposes a new Internet architecture that revolves around the information instead of the host [72]. Some advantages of ICN to the architecture are improved reliability and efficient information/service delivery, making it a promising future networking model for the edge converged 5G applications and systems.

Edge and ICN are necessarily inter-linked and complementary to each other, both of which bring an actual realisation of near real-time 5G application services, although they can function independently [98]. The use of ICN or child concepts such as named data networking (NDN) [99] can enhance the performance of MEC since some challenges existing in highly dynamic MEC ecosystems can be resolved by NDN [100]. For example, one key issue that edge faces is that an application-level reconfiguration needs a service re-initialisation, result in a delay for service migration. Thanks to ICN's ability to simplify the network configuration due to the support of a service-centric approach, which can reduce delays for the reconfiguration [101]. Besides, ICN and NDN can improve the in-network/edge caching and storage for MEC converged networks, which are beneficial for dynamic applications where the same services can be shared by edge devices [102]. Existing works have discussed and proved the importance of integrating ICN with MEC [103], [104]. Researchers in [103] combine MEC and ICN in the connected vehicle use case for traffic information service provisioning. In [104], a converged framework integrating the edge ICN is proposed to enable dynamic orchestration of caching, computing and network resources, enhancing the next-generation vehicular network-related services with substantial delivery improvement. Authors in [98] propose an ICN converged heterogeneous network framework to optimise service availability by allocating virtual edge resources. Ben-Ammar *et al.* [105] propose a strategy that handles the caching of both the service instances as well as their source codes (or software images) while an ICN-based Edge architecture is deployed. Additionally, their strategy does not require any explicit cooperation among nodes that helps to keep the overhead as low as possible.

4) NETWORK SLICING

3GPP is at the forefront for 5G network slicing standardisation initiatives and activities. Network slicing is a key 5G network technology to support tailored and dynamic service availability, enabling high flexibility and scalability. Network slicing can help a physical network to be divided (sliced) into many logical networks to facilitate on-demand customised services for versatile application scenarios using a common and shared physical infrastructure. Additionally, network slicing can allocate network resources to logical slices dynamically and efficiently according to the QoS/QoE requirements. Network slicing integrates the edge/cloud and network resources (e.g., storage, data processing, and RAN access, and required bandwidth). These are also termed edge utilities to fulfilling the dynamic service requirements [48].

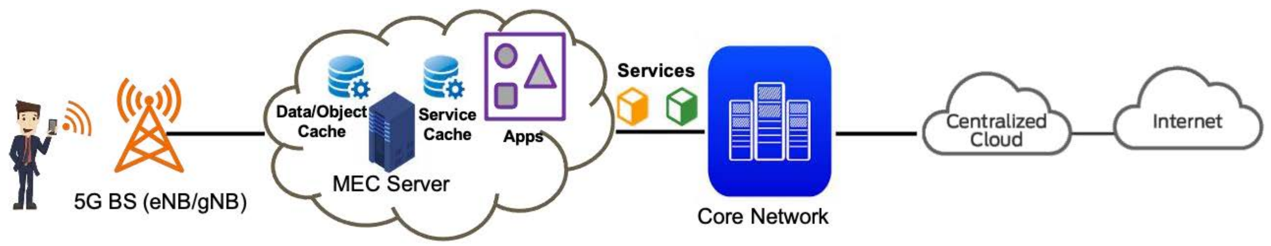

FIGURE 9. Edge caching in 5G MEC.

Network slicing has gained vast research attention recently. The alliance [106] propose network slicing and divide it into three layers: resource layer network, slice instance layer, and service instance layer. Among the layers, the layer related to service instance contains the services (business services or user services) that need to be supported. This layer also handles the network characteristics for a service instance provided by a network slice instance, shared across multiple service instances. Authors in [107], [108] reviewed the literature on 5G based network slicing and softwarization and presented several associated challenges. A 5G architecture based on network slicing is introduced in [109], where the fundamental principles and concepts of network slicing are presented. Authors in [110] present the application of network slicing in the 5G service. [111] present multiple multicast eMBB (enhanced mobile broadband) and URLLC slices in C-RAN and admit the slice requests based on the service provider's requirements. To meet the latency-sensitive service requirements by utilising edge processing, analytics and scalability, the network slicing in MEC is explored in [72]. The use cases include latency-critical services, such as industrial Internet, healthcare, and autonomous driving, elaborated in Section VII. For such delay-sensitive applications, latency minimisation and service prioritisation are two requirements that the edge and network slicing can support. The reasons for convergence are that the service prioritisation can be achieved by network slicing while the edge can attain the latency. Several works have been proposed to combine 5G services and network slicing at the edge for supporting IoT. For example, the challenges related to integrating network slicing with C-RAN and edge are discussed in [112]. Applications in an automobile or massive IoT with network slicing require stringent latency and large scalability, supported by MEC capabilities for such services by providing storage and computation to process/analyse data at the edge.

C. CACHE DEPLOYMENT IN 5G MEC

The 5G network is designed to increase the capacity of cellular networks utilising more bandwidth, a larger scale of antennas, and higher frequency reuse with network densification. However, the demanding network requirements of applications are increasing with a higher speed. For example, VR video streaming will grow 11-fold by 2021, which requires both high throughput (200 Mbps) and low latency (10 ms). Self-driving vehicles connected to the 5G network are equipped with sensing, processing, and communication capabilities that will require a large amount of data, including AR navigation streaming, transportation information, and in-car entertainment. Nonetheless, the backbone Internet faces challenges to provide lower latency and higher bandwidth due to the physical barrier from signal propagation speed, the store-and-forward design, and frequent congestion. To envision a sustainable road map towards future networks, mobile edge caching is essential in the 5G network to provide better QoS for novel applications. Proximity-based service caching and distribution in wireless networks has been identified as a promising service offloading solution for improving both the capacity and the QoE by exploiting service popularity and spatio-temporal requests. There is various caching placement at different places in MEC-Cellular networks, which leads to the deployment of edge caching on servers and within the cellular networks. Service caching in a typical wireless cellular network can be achieved at the core network, RAN, and edge devices [24]. With edge caching enabled service availability, the service-related traffic could be reduced significantly. Figure 9 presents a generic illustration of edge caching in 5G networks.

1) HETEROGENEOUS NETWORKS (HET-NET)

HetNets integrate higher-tier macrocells and lower-tier small cells, such as picocells, femtocells, and relay nodes [76]. The BSs of these cells can be equipped with edge servers, providing storage capacity and CPU/GPU computing for intelligent decision-making. By leveraging the BS-based caching and edge computing system, capacity can be significantly enhanced, and the delay can be significantly reduced [113]. Since these edge servers can implement edge caching by collecting the local service requests in real-time and then use the dynamic information to make intelligent decisions about service reconfiguration with the help of powerful computing ability, including the support for futuristic computationally intensive deep learning algorithms.

- *Macro base station (MBS)*: MBSs are outdoor facilities that can be used for caching in a Het-Net. They usually have a wider coverage area and can serve more users compared to other types of BSs [71]. These characteristics help to boot the service cache hit rate. Some research considers a single MBS with a co-located edge server in their proposed methods [3], [114], while some others [115], [116] consider a collection of a single MBS with multiple small base stations.

- *Small base station (SBS)*: SBSs are expected to be heavily deployed in the next generation of heterogeneous wireless and cellular networks [71]. Caching at SBSs can deliver application services closer to the users, potentially supporting the low transmission power and high data rate [24]. While each SBS is equipped with storage and processing resources, It has a smaller coverage area, limited processing/storage resources, and can support limited users compared to an MBS. In this approach, the local caching gain is obtained when a requested service is locally available in the cache (either at the SBS or the devices) by serving this service from the cache without connecting to the MBS. In some work, the network model comprises multiple SBSs along with an MBS [115]–[118]. However, the researchers in [84], [119] consider the SBSs only. A cache-enabled software-defined Het-Net is introduced in [120]. The CP and the UP are separately configured in 5G RANs. Both the MBS and SBS have cache capability. The MBS contains both CP and UP, while the SBS only has the UP. The service can be pre-cached in the MBS and SBS based on the assessment result in the spatio-temporal domain and local-global views. In the service delivery process, the MBS figures out the suitable SBS for the edge device to connect. The MBS acts as a central controller to distribute the backhaul spectrum resources shared by MBSs and SBSs. The service allocation and distribution is based on the service and data requirements and the SBS functional status.
- *Pico base station (PBS)*: A PBS is a type of SBS that can be used both indoor and outdoor, supporting 32-100 users [24]. PBSs can work in conjunction with MBS, as a central controller, to orchestrate the dynamic service placement. PBSs can work cooperatively to support neighbouring base stations with their cached services.
- *Femto base station (FBS)*: An FBS (also called helper node) is a kind of SBS that usually has low bandwidth backhaul links and is used indoor [24]. FBSs are more cost-efficient for deployment and flexible compared to traditional BSs. Caching at FBSs is referred to as femtocaching [121]. In [122], a single cell is managed by an MBS with several femtocell-like BSs connected to it in a femtocell network. The femtocell-like BSs have weak backhaul links but large storage capacity. The service can distributively be cached in the helpers. Based on the helper density, one edge device can connect to one or more helpers. If the distance among the helpers is considerable, one edge device connects to only one helper, and then the helper caches the services based on the strategy among the connected edge devices for requested services. Generally, services requested by the edge device served by more than one helper can be cached in these helpers separately to minimise the service delay. Several works [13], [46], [84], [123], [124] also use FBSs in their system model.

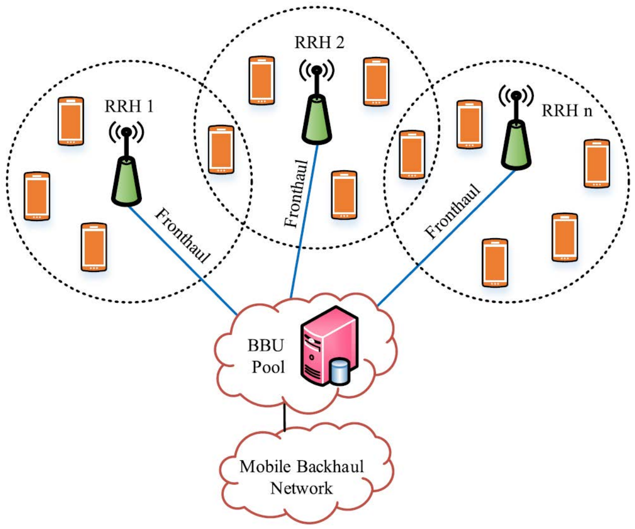

FIGURE 10. The C-RAN architecture [126].

In Het-Nets, users can access the cached services in multiple ways. They can directly access via MSB or SBS (e.g., PBS and FBS). They can also use an SBS as a relay to access the service from MBS. Similarly, they can use MBS as a relay to access the service located at an SBS. Finally, they can use their connected SBS as a relay to access their requested service hosted in another SBS.

2) CLOUD RADIO ACCESS NETWORK (C-RAN)

One of the promising solutions is based on integrating cloud services to radio access networks, called C-RAN, to reduce the CAPEX (capital expenditure) and OPEX (operating expenditure) [24]. In C-RAN (Figure 10), the centralised pool of base band processing unit (BBU) provides the required computational functionalities to the BSs, while a large number of densely deployed low-cost, low-power remote radio heads (RRHs) at cell sites helps improve the network coverage and serve as distributed antennas to interact with users [76], [125]. The fronthaul links connect RRHs to the BBU pools and have constrained capacities.

Caching can address network traffic constraints in both fronthaul and backhaul links, speed up time-critical communications, and provide preferential services. The main caching objectives in C-RANs are to minimise the traffic in fronthaul and backhaul links and the energy consumption of RRHs. In [127], the article discusses the advantages of cloud computing in C-RAN in a software-defined and centralised processing environment and presents a reduced energy consumption with the proposed green and flexible C-RAN. A cloud service-centric networking article [128] introduces cloud computing with RAN caching. In this work, BBU pool based caching is proposed as a service in C-RAN. One of two key aspects is that the BBU provides a much larger

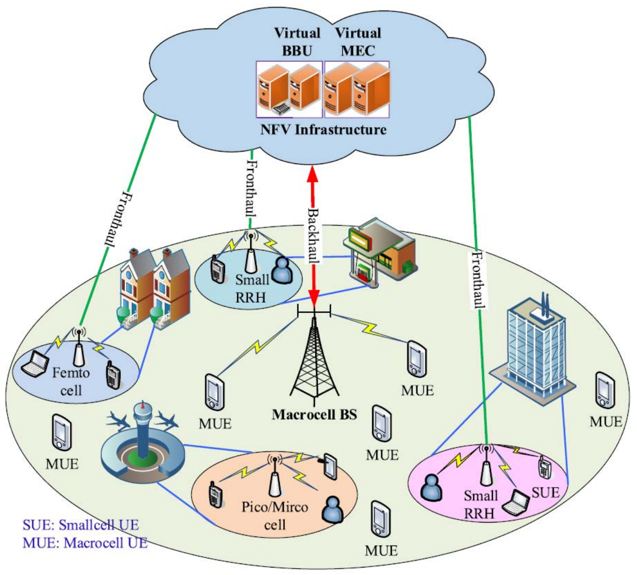

FIGURE 11. H-CRAN MEC architecture [76].

caching size than those in traditional BSs, and secondly, the centralised BBU eliminates the complexity of the distributed service placement. In [129], a cluster oriented caching is proposed in C-RAN. RRHs in the same cluster are served by a local cluster cache at the edge-cloud. The RRHs accessing the same BBU is limited due to the constraint of large cloud implementation complexity in C-RAN. Yin *et al.* [85] introduce a collaborative framework to optimise both data routing and service placement to minimise transferring cost for big data applications in a 5G network. Bhamarea *et al.* [94] introduce an optimal placement scheme in multi-cloud environments for CRANs. Their proposed scheme allocates service demands at base stations in the form of VFs to the VMs to minimise response time while fulfilling various constraints, including cost, capacity, and placement.

3) HETEROGENEOUS C-RAN (H-CRAN)

In the HetNet, low powered edge nodes can lead to severe interference due to the same spectral resources used by MBSs. To avoid this, a heterogeneous C-RAN implementing cloud computing into HetNets (Figure 11) is proposed to gain a scaled cooperative signal processing and networking functionalities [130]. In [131], each BS is equipped with a local cache in the C-RAN, which can be a service-centric RAN. BS caching forms a dynamic cluster. The BS adopts the multicast approach to provide the services and contents for the service delivery mechanism. Combining the edge cache and multicast can improve and satisfy explosive service requests in dense networking environments [76], [132].

D. INSIGHTS

Our work in this survey identifies that several network architectures are proposed and highlighted the constraints in the service placement for low latency QoS and QoE in 5G and beyond. Our survey also shows that some of these architectures can support URLLC service requirements with the aid of MEC, 5G, and heterogeneous access communications in core networks. In order to assure enhanced network efficiency, our analysis sheds light on different aspects (i.e., airside, network edge, and core network layers) of integration solutions for dynamic services and applications. Although, standardisation efforts and potential technologies are discussed to address the in-network placement problem to establish an end-to-end working MEC ecosystem, including but not limited to: (i) use of HetNet in 5G to allow high QoS and QoE overcoming the resource and sharing constraints. (ii) employing centralised and distributed control and data planes for differentiated service and technology models to facilitate interaction between network and applications. However, we find it still lacks at the fronts of a comprehensive study to investigate a global optimal in-network service placement. We learn from our work that different service-enabling network architectures for a potential MEC deployment considering SDN/NFV, ICN and network slicing can enable distinct and flexible network orchestration. Our survey compares several studies on service cache deployment locations to address sub-optimal service placement in cellular base stations including MBS, SBS, PBS, FBS, and C-RAN with their corresponding challenges and benefits to support dynamically changing requirements. However, an advanced integration of the MEC paradigm in 5G and beyond, given the relative infancy of the field for the dynamic service placement, requires further research to support new applications and their micro- requirements/dependencies.

VI. MIDDLEWARE LAYER PERSPECTIVE

A middleware is a software layer that provides a system-wide transparent view to application developers by abstracting the complexities of heterogeneity, communication network, and computation concerns [133]. The middleware services interact with operating systems, network communication layers, and applications to facilitate coordination and collaboration among user devices, edge servers, and the cloud infrastructure. The middleware can offer services responsible for the configuration of application services (service instances) in edge servers for performing dynamic placement. In this section, first, we investigate the influential characteristics of different types of service instances. Then, we analyse current research directions in dynamic service placement. Next, we propose a taxonomy of design objectives for these works, and finally, we introduce our devised taxonomy of dynamic service placement solutions.

A. SERVICE INSTANCE

Application services are usually packaged in the form of service instances using packaging technologies. The packaging technologies have different characteristics that impose limitations and opportunities (trade-offs) for a candidate dynamic service placement solution. Before discussing these trade-offs, we need to review these characteristics and the available packaging technologies.

1) SERVICE PACKAGING CHARACTERISTICS

We identify four characteristics of isolation level, platform independence, footprint size, and launch time [39], [48], [53].

a: ISOLATION LEVEL

An isolation level refers to the level of vulnerability within the services of an application and the vulnerability level across different applications [53], [134]. In the former case, a low isolation level causes the proliferation of software bugs and failure [39], and in the latter case, it exposes security and integrity threats [53], [134]. The isolation levels can be divided into five classes of system-level, kernel-level, process-level, and thread-level. For example, system-level isolation indicates that a failure caused by an error will not affect the correctness or sometimes the performance of other programs running on the same host machine but in a different running environment such as a different VM.

b: PLATFORM INDEPENDENCE

A MEC environment encounters various forms of heterogeneity, including hardware infrastructure, OS platform, OS version, or required libraries. The heterogeneity implies new requirements regarding the level of independence of service instances from an environment that runs the services. For example, container technology usually operates in an identical OS platform (OS-level independence) while providing flexibility on the OS version.

c: FOOTPRINT SIZE

This characteristic refers to the actual number of bytes for a service instance. In general, the footprint size of a service instance is affected by several factors such as functionality (lines of code) and embedded data/datastore. However, a packaging technology may require additional components to manage resource dependencies and the life-cycle of a service instance. The footprint size has significant effects on the performance of service placement solutions in terms of disk storage, memory, and bandwidth usage due to frequent swap ins/outs [135].

d: LAUNCH TIME

It is the time to transfer a service instance until the instance becomes ready to serve service requests since the instance requires some time to boot and get ready to fulfil service requests (demands) after being transferred to a destination edge server. The interval between service instance arrival and readiness to serve is called *Launch time*. In the literature, it is also referred to as *booting-up time* [136], [137], *start-up activation* [138], or *instantiation delay* [95]. A short launch time will boost the throughput of edge servers, which is important in dynamic service placement solutions.

2) SERVICE PACKAGING TECHNOLOGIES

We investigate the following service packaging technologies:

a: VIRTUAL MACHINES

System-level virtualisation, usually called a *virtual machine* (VM), is one of the enabling technologies in the cloud and edge computing ecosystem. The technology allows MEC and cloud platforms to share physical computing resources (i.e. processing, storage, memory) among users [53]. VMs run on top of virtual machine monitors (such as Hypervisor) and use virtualised hardware resources provided by a host machine. Each VM hosts a full-flagged operating system that leads to hardware-level independence as well as system-level isolation. The highest isolation level comes in the cost of duplicating what is offered by the host operating system. As a result, the footprint size of VMs is large, which is an important obstacle in widely employing them in the dynamic service placement of MEC. Moreover, hypervisor-based virtualisation exhibits considerable processing overhead and high launch time [138].

b: CONTAINERS

The monolithic view for designing mobile cloud applications hinders the high demand components of these applications from scaling separately. An alternative is to design separated and loosely coupled services that can communicate with each other. These services are packaged in the form of service images and executed in various edge servers as containers. Containers consist of an application, libraries, and other dependencies such as a file system. They provide application portability without software installations [139]. Containers achieve kernel-level isolation [53], which implies a container crash/compromise may affect other containers on the host machine. Unlike VMs, containers have a strong dependency on the host operating system kernel. They share more resources of the host operating system in common [11], [39], such as their embedded libraries and the local file system. On the one hand, sharing common resources helps them have a smaller footprint size than VMs, allowing hundreds of containers to be hosted on a physical machine. On the other hand, the portability must be between compatible operating systems that assure the particular dependency. The containerisation also has lower system overhead compared to VMs and benefits from faster launch time [116]. Wang *et al.* [39] introduce several advantages of using containers. To name a few, container abstraction reduces complexity, containers support automation, and higher computing capability can be provisioned because a service can be split into several separate containers.

c: PROCESS VIRTUAL MACHINE

A process virtual machine is a runtime system that essentially provides an abstract instruction set for executing an application [47], [134]. For instance, *Java virtual machine* (JVM) allows the offloaded Java compiled bytecodes to

TABLE 7. The comparison of different service instance packaging technologies.

Service instance type	Isolation level	Independence	Footprint size	Launch time
Virtual machine	System-level	Hardware-level	Large	Slow
Containers	Kernel-level	OS-level	Medium	Moderate
Process virtual machine	Process-level	Runtime environment	Small	Fast
Agents	Thread-level	Agent platform	Small	Fast
Functions	Thread-level	System libraries	Very small	Fast

be executed in any machine containing *Java runtime environment* (JRE).

d: AGENTS

A software agent is an autonomous computer program that can interact with its environment [140]. Mobile agents are software agents which can travel from one agent platform (execution environment) to another using computer networks [141], [142]. The mobility of the application code provides dynamic reconfiguration in both application components and the application's location. The agents are capable of cooperation, social interaction, and collaboration with other agents. They also can perform appropriate actions towards their goal reactively or proactively. The above properties make agents a good candidate for service instance packaging [143], [144]. The agents require an identical execution (runtime) environment that imposes less processing overheads and a smaller footprint size on edge servers compared to VMs. *Java agent development framework* (JADE) is an example of an agent platform that simplifies the implementation of multi-agent systems through middleware. *JADE* is fully implemented in the Java language that empowers the developed systems to be distributed across machines with heterogeneous operating systems as long as they are equipped with the *JADE* platform. However, agent-based programming is a fundamentally new programming paradigm to software developers compared to functional or object-oriented programming.

e: FUNCTION

Applications can be built up using a set of dependent or independent functions invoked by a single service call or an event. The idea of using functions as service instances emerges from serverless computing [139], [145]–[147], also known as *function-as-a-service* (FaaS) [53]. The objective is to provide a cloud computing model for stateless and event-driven applications, where application functions will be executed when necessary without consistent execution of the entire application [148], [149]. Examples of platforms that currently support this computing model include Amazon *AWS Lambda*,[1] IBM OpenWhisk,[2] Google Cloud Functions,[3] *Knative*, *OpenFaaS*, and Microsoft Azure Functions.[4] To review a quantitative and qualitative evaluation of serverless computing frameworks support at the edge, the interested readers may refer to Palade *et al.* [149].

3) DISCUSSION ABOUT TRADE-OFFS

Table 7 presents the comparison of different service instance packaging technologies. We pinpoint three prominent trade-off cases. The first case is the heterogeneity of edge servers due to the presence of multiple edge server operators in which each operator employs a different edge server platform from another operator. The heterogeneity of edge servers requires using VMs that provides system-level isolation and hardware-level independence; however it comes with the cost of large footprint size and a slow launch time. A dynamic service placement solution has to limit the number of swap ins/outs or prolong service caching decisions to mitigate the drawbacks. In the next case, there is a consensus among edge server operators on using a similar OS platform. By accepting this constraint, they can achieve several advantages such as a smaller footprint size of service instances and quicker launch time which introduce flexibility in the number of swap ins/outs and better adaptation to the changes in service demand pattern without deteriorating the overall system performance. Containers can be a popular choice for this case which provide a kernel-level isolation level. If the edge server operators accept more constraints and use a common agent platform, then using mobile agents can also be an option. Finally, the last case describes a situation where a small footprint size and quick launch time are the most dominating requirements. The price of fulfilling these requirements will be paid by lowering the isolation level to thread-level isolation and the independence level of the system libraries.

B. RESEARCH DIRECTIONS

The conducted research in dynamic service placement problem can be classified into several research directions. These directions affect problem assumptions, system model and introduce diverse research challenges. We identified eight research directions according to our surveyed papers. Table 8 shows the results of our analysis.

1) CONTEXT-AWARENESS

The context is a useful piece of information that offers new opportunities in adapting a service placement method to system dynamics. According to Dey *et al.* [211] definition:

> *any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the*

[1]https://aws.amazon.com/lambda/
[2]https://developer.ibm.com/openwhisk/
[3]https://cloud.google.com/functions/
[4]https://azure.microsoft.com/en-us/services/functions/

TABLE 8. Service placement research directions.

Work	Context-awareness	Budget constraint	Coverage overlap	Service heterogeneity	Edge server heterogeneity	Service inter-operation	Privacy/ Security	Mobility model
[113]	Both	-	✓	✓	✓	-	-	-
[150]	Context-free	✓	✓	✓	✓	-	✓	-
[83, 15, 117, 22, 23, 45]	Context-free	✓	✓	✓	✓	-	-	-
[151]	Context-free	✓	✓	✓	-	-	-	-
[152, 153, 154]	Context-free	✓	-	✓	✓	✓	-	-
[84]	Context-free	✓	-	✓	✓	-	✓	-
[87]	Context-free	✓	-	✓	✓	-	-	✓
[155, 156, 157, 158, 159, 119]	Context-free	✓	-	✓	✓	-	-	-
[160]	Context-free	✓	-	✓	-	-	-	✓
[161, 162, 163]	Context-free	✓	-	✓	-	-	-	-
[164]	Context-free	✓	-	-	✓	-	-	-
[165]	Context-free	✓	-	-	-	-	-	-
[124, 166, 167]	Context-free	-	✓	✓	✓	-	-	✓
[86, 13, 168, 169, 170, 94, 171, 172, 95, 173, 174, 175, 176]	Context-free	-	✓	✓	✓	-	-	-
[177]	Context-free	-	✓	✓	-	-	-	✓
[114, 4]	Context-free	-	✓	✓	-	-	-	-
[178, 179]	Context-free	-	-	✓	✓	✓	-	-
[93, 180]	Context-free	-	-	✓	-	✓	-	-
[181, 182]	Context-free	-	-	✓	✓	-	-	✓
[85, 183, 184, 185, 137, 186, 81, 3, 187, 188, 189, 190, 191, 105, 192, 193, 96]	Context-free	-	-	✓	✓	-	-	-
[10, 194, 195, 196, 197, 118, 198]	Context-free	-	-	✓	-	-	-	-
[199, 200]	Context-free	-	-	-	✓	✓	-	✓
[201]	Context-free	-	-	-	✓	-	-	-
[123, 202, 203]	Context-free	-	-	-	-	-	-	-
[116]	Context-aware	✓	✓	-	✓	-	✓	-
[14]	Context-aware	✓	-	✓	✓	-	-	✓
[115]	Context-aware	✓	-	✓	✓	-	-	-
[204, 82]	Context-aware	✓	-	-	-	-	-	✓
[205, 206]	Context-aware	-	✓	✓	✓	-	-	✓
[207]	Context-aware	-	✓	✓	-	-	-	✓
[208]	Context-aware	-	-	✓	-	-	✓	-
[209]	Context-aware	-	-	✓	✓	✓	-	✓
[210, 46]	Context-aware	-	-	✓	✓	-	-	✓

interaction between a user and an application, including the user and applications themselves.

These entities can be described by several attributes [212] such as *identity*, *location*, *status*, and *time*. The interpretation of the definition and related attributes in dynamic service placement suggests the following four entities and attributes.

- *Mobile user entity* refers to the user that requests a mobile cloud service and may include a wide range of attributes. The first group of attributes are demographics such as age, gender, nationality, occupation, and education [82], [116], [204], [208]. These attributes help dynamic service placement solutions to categorise mobile users and discover the service demand pattern of each category. Users locations are the next group of attributes [14], [82], [113], [204], [206]–[210] that are used to enhance the prediction of service request loads and the target area of the users' movement. The location information can be the cell where the user is located, the GPS information, or a tuple of (Position, Position error, Speed, Speed error, Direction, Direction error). Time is the next attribute of the mobile user entity [115], [116] that is exploited to uncover service demand pattern either in combination with other context information or solely. Finally, several works [82], [116], [204], [205] consider user preferences. The preferences can be declared explicitly [205] by mobile users or inferred [82], [116], [204] based on demographic attributes or users' social information.
- A *user device* is equipment that the mobile user uses to request a service. Two main attributes of this entity are device type (i.e. smartphone, tablet, gadget) and device status, such as battery level [82], [116], [204].
- An *edge server* refers to a computing node near a base station used for executing mobile cloud services. While the population of users supported by the edge server is an edge server attribute, the status of edge servers provides valuable context information, especially for distributed dynamic service placement solutions.
- An *external event* refers to occasions such as concerts that influence service demand patterns. The location and time are the main attributes of this entity [115], [116].

The context information may include sensitive user information such as locations and requested services that raise concerns about distributing them among other edge servers or the cloud due to the exposure of privacy violation. Consequently, it is necessary to provision privacy protection mechanisms [212], [213]. For more information about context-aware systems, interested readers may refer to [214].

2) BUDGET CONSTRAINT

Several works consider budget constraints in their system models. They fall into one of the following categories.

The first category of research [45], [87], [116], [155], [157], [159], [204] assumes that ASPs have a limited long-term budget for dynamic services caching. This budget is defined based on the service hosting cost or the consumed resources (i.e. bandwidth [204], backhaul usage [15], the energy consumption of network nodes [87]) by the cached services. The limitation imposes a budget constraint on the cost of each service placement which adds a barrier for ASPs to cache their services in an excessive number of edge servers. Additionally, the limitation hinders frequent reconfiguration of services in edge servers. Zhou *et al.* [22], [23] introduce a budget-constrained dynamic service deployment method in which the cost model comprises service installation cost, computation cost, and traffic cost. The service installation cost refers to the expenses of service copyright and maintenance, computation cost corresponds to the utilisation of computational resources in the edge servers, and finally, traffic refers to the expenditures paid for incoming and outgoing data transmission.

In the second category [83], [150], [156], [158], ASPs have to rent computational resources of edge servers from edge server operators (network operators, small-cell base stations [115]). The computational resources are generally in the form of CPU power, memory, or storage resources. ASPs are charged based on the number of requested computation resources [115] or their rented time slots in an edge server. ASPs have to balance their revenue with the rental price of edge server resources. Some research focuses on maximising profit [119], [154], [163]. From another viewpoint, rental prices can be fixed [15], [117], [161], [215] or dynamic [150]. In the latter case, the price is determined according to the time and location of an edge server. One of the common approaches in dynamic pricing is to use an auction mechanism; for instance, a forward auction is used in [150], or *Vickrey-English-Dutch* (VED) auction is applied in [160]. The game-theoretic modelling is another alternative approach in dynamic pricing used in some works, including [83], [84], [151]. Renting edge server resources also depend on the utilised technology, namely Infrastructure as a Service, Containers as a Service, and Function as a Service, where the rental price can follow cloud providers' billing mechanisms [215]. Liu *et al.* [164] study the incentive mechanism and introduce an optimal payment strategy based on the Stackelberg game.

3) COVERAGE OVERLAP

In some network architectures, such as small-cell networks, the communication coverage of base stations may overlap due to the dense deployment of small base stations. Several works consider this coverage overlap in which a user device can be served by more than one base station. However, it is usually assumed that the device is connected to only one of them at each point in time.

In coverage overlap research direction, researchers have to address two design issues.

- Considering a user device within coverage areas of multiple base stations, the first design issue is to decide whether user devices are in charge of selecting a base station or the base stations have to select their serving user devices. Some researchers in [22], [23], [171] put the burden on user devices by selecting a base station according to the quality of its wireless channels, while some others [113], [168] consider that the base stations can select user devices based on the number of occupied and available channels.
- The second design issue is to identify the possible implications of using coverage overlap. The coverage overlap can be used for wireless channel management [22], [23], [171]. Besides, a user device may have direct access to the cached services of multiple base stations within its reach.

In the presence of coverage overlap, service demand and communication/computation resources become highly coupled [13], and overlapping base stations have to collaborate among themselves. For instance, base stations can cooperatively decide which services to host [113], [168]. Chen *et al.* [116] model the small-cell network as a graph where base stations represent vertices, and the edges connect the overlapping base stations. Then, they defined a *graph component* as a sub-graph in which any two vertices are connected by paths. They proposed *SEEN-O* method in which service placement decisions is made component-wise instead of an individual base station decision.

4) SERVICE HETEROGENEITY

The significant diversity of application services in the MEC environment imposes dynamic service placement solutions to consider the heterogeneity of services in their system models. We categorise this heterogeneity into three main areas.

- Application services need various amounts of computational resources. The heterogeneity in required processing power (maximum required CPU cycles) is considered in many works. Disk storage is another source of heterogeneity, especially in VM-based service instances in contrast to function-based technologies (such as serverless computing) in which the importance of disk storage is blurred. Finally, memory is the last computation resource that is taken into account.
- The second area is the variation in service level agreements (SLA) and QoS requirements which are generally based on time-related factors. AR [190], online gaming [170], [199], [216], safety-related services in autonomous/cooperative driving [139], [217]–[219], and health services [220] are among the applications that dictate stringent QoS requirements in terms of response time and execution deadline.
- Application services have different input/output parameter sizes that imply diversity in upload/download data

sizes. From the networking perspective, the required network bandwidth depends on the summation of these data sizes. The required up-link and down-link traffic can also be studied separately.

To handle the service heterogeneity, application services can be classified according to their similar characteristics regarding their required resources and QoS requirements. Sonmez *et al.* [167] group the application services into four main types of AR, health applications, compute-intensive, and infotainment applications. The application types have different delay sensitivity, upload/download data size, and task length in terms of processing power.

5) EDGE SERVER HETEROGENEITY

Most of the dynamic service placement methods assume edge servers with different computational resources. The main areas of heterogeneity are processing power, disk storage, and memory ordered by their level of importance. He *et al.* [184] classify these resources into shareable and non-shareable types. According to their definition, if a resource unit can serve more than one unit of service demand, it is considered shareable. For instance, disk storage is a shareable resource since the stored service instance can serve multiple demands of the same service. In contrast, non-shareable resource unit, such as CPU, can serve only a single demand. The optimal provisioning of application services with non-trivial demands of shareable and non-shareable resource types is introduced in [184].

6) SERVICE INTER-OPERATION

Mobile applications are comprised of a set of services. Additionally, a mobile application needs to interact with different services to accomplish its goal. The inter-operation between the services can be based on an ordered sequence of service requests (a service chain), a workflow, or a complex graph of service requests. An ideal dynamic service placement solution has to consider the inter-operations between these services since relocating a service with significant data exchange with co-located services may reduce the performance due to excessive data transmission over the network. There are two scenarios to study this research direction.

In the first scenario, the inter-operation is assumed between application components/services within user devices [41]. Several works address the question of which application service(s) should be offloaded from a user device to an edge server [17], [18], [221]. Some of the influential factors are delay sensitivity, required computational resources, size of exchanged data with the offloaded service, and the size of offloading code.

In the second scenario, the service inter-operation is assumed between services that are placed outside user devices. Several use cases of this scenario are studied in the literature [93], [178], [180], [199], [200], [222], [223].

- The chaining of network functions is extensively studied in recent years [224], [225]. While many of these methods proposed static chaining, some others, such as [152], [226], [227], introduced dynamic chaining to meet QoS requirements and adapt to network dynamics.
- The second case is the service placement in Fog computing [62], where applications follow the *sense-process-actuate* model [187]. The main goal is to provision the optimum placement of multiple components of an application given their decencies. In the static service placement case, the service dependency can be modelled as a direct acyclic graph [228]–[231]. For the dynamic scenario, Guerrero *et al.* [178] consider stateless inter-operating microservices in which at least one service instance is hosted in the cloud. They introduced a service placement method in which services are placed as close as possible to the clients' gateway. To address the resource limitations of nodes, a service can migrate to other nodes along the shortest path from the cloud to the gateway. In another work, Chiti *et al.* [93] introduce a virtual function placement method. The chaining of virtualised functions is also referred to as *service function chaining* (SFC), where the deployed applications are considered as a set of independent services, cooperating in a typically sequential order following sense-process-actuate workflow.
- The last case considers service dependency in a MEC environment. Bahreini *et al.* [209] address the multi-component application placement problem. Zhao *et al.* [179] consider microservice-based applications with sequential combinatorial structure. Peng *et al.* [153] assume that applications can be in the form of a divisible service chain. Prakash *et al.* [154] introduce a service caching method that decides what fraction of the service to cache. Another example is a social VR application in which multiple service entities located in edge servers. Each service entity is associated with a user (client entity) and is responsible for computation-intensive tasks and tracking the interaction between users. Service entities have to frequently exchange metadata with client entities and other service entities while users are interacting with each other. Wang *et al.* [199], [200] address the *edge service entity placement* problem by encoding all the costs (including activation cost, the placement cost, the proximity cost, and the colocation cost) as a graph. Then, they converted the cost optimisation into a graph cut problem.

To summarise, most of the dynamic service placement works in MEC environments do not consider the inter-operation between the application services. However, some of the above-mentioned methods can be applied in MEC.

7) PRIVACY/SECURITY/TRUST

Several characteristics of a MEC environment raise concern regarding privacy, security, and trust issues. First of all, the MEC environment can store users' locations and their requested services [20]. Secondly, edge servers can be owned by third parties [48]. Finally, dynamic service placement methods may share service demand statistics with entities that make service placement decisions. There are several best practices regarding privacy, security, and trust.

To preserve user privacy, Qian *et al.* [208] use a distributed federated learning method to learn service preferences of the users. The method helps to prevent users' sensitive information from sharing with other network entities. Preserving user privacy is of significant concern in context-aware placement methods. Chen *et al.* [116] introduce a context-aware method in which small base stations organise context space depending on their local users' privacy preferences.

To maintain security concerns, Luo *et al.* [150] use a security level and access capability for both edge servers and users in an industrial environment where users can offload their tasks to edge servers that match the security level.

Chen *et al.* [84] introduce a social trust model between small base stations (SBS) since individual owners operate them. The SBSs can form a coalition under this model that facilitates offloading tasks from one SBS to another SBS.

8) MOBILITY MODEL

User mobility is an intrinsic characteristic of the MEC environment. In considering user mobility, two main aspects need to be answered.

The first one addresses the way to capture user mobility. A subset of works in dynamic service placement utilised a specific mobility model, a real-world mobility trace dataset, or a synthetic dataset. The researchers in [87], [205] apply the mobility models of *ONE simulator*. A few works use a specific mobility model; for instance, *smooth random mobility model* [124], *random walk* [209] and *nomadic mobility model* [167] are used in the literature. Real-world mobility traces are also considered in several works, for instance, Shenzhen city taxi and metro dataset in [14], mobile users of Twidere dataset in [82], [204], Rome taxi dataset in [199], Shanghai taxi trajectory dataset in [166], real-world check-ins in [207], and mobility of flights in [182]. Finally, the authors in [160] use a synthetic dataset generated according to the region around the city of Köln. It is worth pointing out that modelling human mobility is still a topic of research [232]. Another alternative to express user mobility is to combine users mobility with service requests models (task arrival models) by applying probability distribution functions such as *Poisson distribution* [171], [201], [215], [233].

The second aspect of user mobility explains how to analyse users mobility data to predict their trajectory/destination, which facilitates the prediction of service demand patterns [124], [204], [206]. Ouyang *et al.* [87] argue that an accurate prediction of user mobility over the long run is extremely hard due to the unpredictable nature of human mobility. Some methods, such as [206], use base station cells as user locations instead of actual GPS locations to handle the inaccuracy.

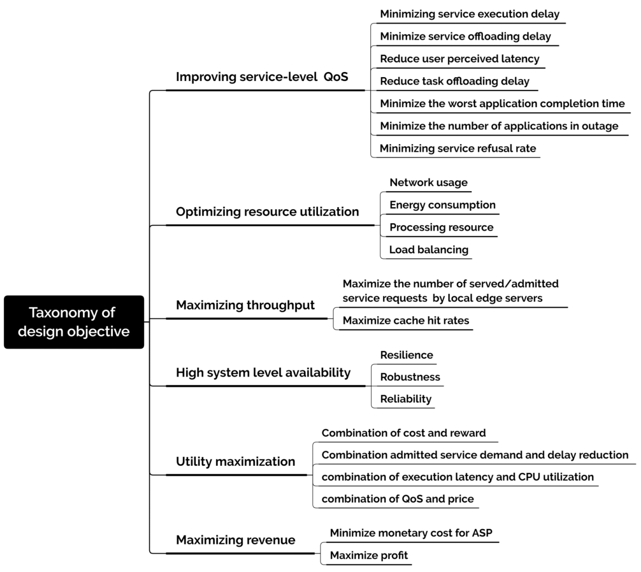
Taxonomy of design objective
Improving service-level QoS
Minimizing service execution delay
Minimize service offloading delay
Reduce user perceived latency
Reduce task offloading delay
Minimize the worst application completion time
Minimize the number of applications in outage
Minimizing service refusal rate
Optimizing resource utilization
Network usage
Energy consumption
Processing resource
Load balancing
Maximizing throughput
Maximize the number of served/admitted service requests by local edge servers
Maximize cache hit rates
High system level availability
Resilience
Robustness
Reliability
Utility maximization
Combination of cost and reward
Combination admitted service demand and delay reduction
combination of execution latency and CPU utilization
combination of QoS and price
Maximizing revenue
Minimize monetary cost for ASP
Maximize profit

FIGURE 12. Taxonomy of design objectives.

C. TAXONOMY OF DESIGN OBJECTIVES

We classify the design objective of dynamic service placement methods into six groups. Figure 12 shows our proposed taxonomy of design objectives.

1) IMPROVING SERVICE-LEVEL QOS

One of the common design objectives is to boost the QoS. Several works targeted reducing service delay/latency, including minimising service execution delay and offloading delay. Some of the works concentrated on improving the QoE by lowering user-perceived latency. Moreover, the design objective of some works is to minimise the worst application completion time and reduce the number of applications that their completion time is greater than a predefined deadline (applications in an outage). Finally, the researchers in [152] define *service refusal rate*, which is defined as the ratio of the number of refused service graphs (chain) and the number of orchestrated ones while the CPU utilisation of the whole network is below a predefined threshold in 90% of the time. Their objective is to minimise the service refusal rate.

2) OPTIMISING RESOURCE UTILISATION

Resource usage optimisation is a design objective in a significant number of works. The design objective includes network usage, energy consumption, and processing resource.

- Minimise network usage: The main idea is to reduce the backhaul traffic by processing service requests in nearby edge servers. There is a trade-off between using the backhaul traffic to download a service instance (for caching) and saving backhaul traffic by preventing to forward the service requests to the cloud. Several works aimed at reducing the traffic of forwarding requests to the cloud and reducing the traffic for downloading service instances from the cloud. The objective is to restrict the number of cache switching, leading to reduced caching (transition) cost of service placement.

- Minimising energy consumption: There are two objectives regarding energy consumption. The first one is to minimise user devices' energy consumption, which is primarily considered in computation offloading scenarios. The second objective, which has received less attention, is to minimise an edge server's energy consumption by considering a long-term energy constraint.
- Maximise processing resource utilisation: The goal is to maximise the exploitation of edge server resources, especially the CPU resource. While some works follow this objective, it is worth noting that workload orchestration has to be considered alongside this objective to prevent a case where an edge server becomes overloaded in the vicinity of an under-loaded one.

3) MAXIMIZING THROUGHPUT

The goal is to maximise service processing rates in edge servers, which can be achieved either by maximising the number of served/admitted service requests by local edge servers or maximising cache hit rates.

4) HIGH AVAILABILITY

Application services may fail to serve users due to runtime failures or transient network interrupts. The heterogeneity of edge server platforms also increases the risk of failures, while some failures are inherited from the environment, such as disaster relief ambiences. Multiple research aims at high availability [179], resilience [234], robustness [235], and reliability [236] to ensure service reliability and continuity.

5) UTILITY MAXIMIZATION

Some of the works pursue multiple objectives, and they model these objectives as a utility function. The design objective of these works is to maximise the modelled utility function, including cost and reward [13], [84], combination of QoS and price [114], combination admitted service demand and delay reduction [115], [116], and combination of execution latency and CPU utilisation [210].

6) MAXIMIZING REVENUE

This objective aim at reducing the costs as well as getting the highest revenue. Edge server rental is one of the primary cost sources for ASPs. The cost has to be covered by serving an excessive number of service requests by the edge server. The second cost source is related to the number of requests not served by an edge server and has to be served by the cloud at a higher cost. A few works target maximising revenue/profit by focusing on cost and revenue.

Tables 9 and 10 present the design objectives of surveyed papers and their addressed sub-problems.

D. TAXONOMY OF SOLUTIONS

We investigated the dynamic service placement solutions and identified seven features that are further used to propose our taxonomy of solutions (Figure 13).

1) SERVICE INSTANCE TYPE

Dynamic service placement solutions have to consider the strengths and weaknesses of service instance packaging technologies. A minor group of works declared their target service instance type. These works mostly considered VMs technology; however, there is a growing trend in using containerisation technologies in recent works. In the third place, the researchers utilised functions as a service.

2) CENTRALISED VS DISTRIBUTED

In a centralised scheme, the placement decision for all or a group edge servers occurs in a single deciding entity. This entity can be the back-end cloud, a centralised edge orchestrator, or an ASP. The central entity has a global view of the service demands, network status, and available resources in each edge server which help to achieve optimum or near optimum performance. However, the performance comes with the cost of constant updating of the network status, service demands, and available resources that are challenging in a highly dynamic environment.

In the distributed (decentralised) scheme, each edge server is responsible for its placement decisions according to its local view. The nearby edge servers can also share their local view to perform distributed cooperative placement.

A hybrid scheme is used in some works such as [87], [173], [206], [208]. One strategy is to partially process the required placement information in edge servers and forward the results to a central entity to make the placement decision. For instance, Wei *et al.* [206] use edge servers for mobility prediction. A central entity predicts the number of future service requests using a back propagation neural network model. Qian *et al.* [208] combine a centralised greedy method and distributed learning method for service placement. Another strategy is to process the global network-wide information in a central entity and then use a distributed approach to make the placement decisions. For example, *Elbamby* in [173] introduces a hybrid approach in which a centralised controller is used to cluster end-user nodes based on their spatial proximity and mutual interest in popular tasks. Then, distributing the tasks among cloudlets is performed by a distributed matching game algorithm. Ouyang *et al.* [87] propose a hybrid approach that used the Lyapunov optimisation technique to incorporate the long-term budget into a series of real-time optimisation problems. Then, the users apply a greedy heuristic to adopts the best response to optimise their own placement decision.

3) COOPERATIVE VS UNCOOPERATIVE

The edge servers have limited resources. Designing an independent placement decision for each edge server results in

TABLE 9. Design objectives and covered sub-problems.

Work	Service caching	Resource allocation	Request forwarding	Design objective
[85]	✓	✓	✓	Minimise data transferring cost
[83]	✓	✓	✓	Maximise profit
[178]	✓	✓	-	Reduce network usage and service latency
[123]	✓	-	-	Minimise the total caching cost while satisfying requirements of user utility which is the sum of the gained utilities by each client from the cached entities
[114]	-	✓	✓	Maximise the total revenue, minimise the service latency, and high QoS requirement
[150]	✓	✓	✓	Maximise profit
[205]	-	✓	✓	Minimise user's perceived latency
[177]	-	✓	✓	Reduce delay and energy consumption
[86]	-	✓	✓	Reducing the network latency in processing the offloaded tasks
[10]	-	✓	✓	Minimise total cost (forwarding requests and downloading services)
[115]	✓	✓		Utility maximisation
[13]	✓	-	✓	Utility maximisation
[168]	✓	✓	✓	Minimise the traffic load caused by service request forwarding
[93]	✓	✓	-	Minimising the worst application completion time and the number of applications in outage
[14]	✓	✓	-	Minimise user-perceived latency, resource usage cost, and service placement transition cost
[15]	✓	✓	✓	Minimise offloading cost including user energy consumption, service caching cost, and the cloud usage cost
[117]	✓	✓	-	Maximise the latency reduction for mobile devices and minimising the monetary cost for ASP
[183]	✓	-	✓	Minimise communication latency
[124]	✓	✓	✓	Reduce task offloading delay
[113]	✓	-	✓	Minimise traffic load to the cloud and minimise service switching cost
[22]	✓	✓	✓	Minimise service response latency and keep operation costs bellow the budget
[169]	✓	-	-	Minimise cost (generic cost function)
[170]	✓	-	-	Minimise service and switching cost
[151]	✓	✓	✓	Minimise resource rental costs
[160]	✓	✓	-	Improve QoS (application latency)
[94]	✓	✓	-	Reduce the total delay
[180]	✓	-	✓	Minimising energy consumption
[152]	-	✓	✓	Minimise bandwidth and deployment costs and minimising service refusal rate
[87]	-	✓	-	Minimise user-perceived latency
[184]	✓	✓	✓	Serving the maximum user requests and minimise the cost in serving all the users
[185]	✓	✓	✓	Minimising execution delay and energy consumption in computation offloading
[171]	✓	✓	✓	Minimising computation latency under a long-term energy consumption constraint
[172]	✓	✓	✓	Maximises the number of requests served by the BSs
[201]	-	✓	✓	Minimising network latency
[206]	✓	✓	✓	Maximise the number of locally served users
[137]	✓	✓	-	Maximising the utilisation of processing resource and maximise the QoS experienced by users
[186]	✓	✓	✓	Reduce latency and energy consumption
[204]	✓	-	✓	Minimise the long-term time average service delay under the long-term cost budget constraints
[208]	✓	✓	-	Protect users' privacy and provide better QoS to users by decreasing the load of services on the remote cloud
[181]	✓	✓	✓	Service quality meets users' expectations
[194]	✓	✓	✓	Minimise total cost including including the cost of forwarding requests and downloading services
[81]	✓	✓	-	Maximise cache hit rates
[23]	✓	✓	✓	Minimise the average latency
[199]	✓	-	-	Minimise the total cost (activation, placement, proximity, and Co-location costs)
[3]	-	✓	✓	Minimise energy consumption
[155]	✓	✓	✓	Maximise the percentage of served requests under budget constraints
[4]	✓	-	✓	Reduce service latency
[161]	✓	✓	✓	Maximise total revenue
[116]	✓	✓	-	Maximise the expected utility
[187]	✓	✓	✓	Maximise the utilisation
[166]	-	✓	✓	Improving QoS (access delay, switching delay and communication delay)
[167]	✓	✓	✓	Maximise resource utilisation
[188]	✓	✓	✓	Minimising the total backhaul traffic
[189]	-	✓	✓	Efficient computational offloading
[84]	-	✓	✓	Increase system utility
[190]	-	✓	✓	Minimise the overall service latency maximise the total analytics accuracy of the mobile AR
[95]	✓	✓	✓	Maximise the number of requests admitted while minimising their admission cost
[195]	-	✓	✓	Minimise energy consumption
[173]	✓	✓	✓	Minimise the total task computing latency under reliability constraints
[207]	✓	✓	-	High availability, Minimise the response time
[209]	✓	-	✓	Minimise the cost
[174]	✓	✓	✓	Maximise the efficient use of resources, Prevent SLA violations
[164]	✓	-	-	Maximise the utilities of cloud service operator and edge server owners
[156]	✓	✓	✓	Multi-objective (deadline violation, operational Cost, and unavailability)
[191]	✓	✓	✓	Minimise the response time
[182]	✓	✓	✓	Minimising the total cost
[196]	✓	✓	-	Minimise the total latency of arrived tasks in a long term
[197]	✓	✓	-	Maximise the long-term average reduction of delay
[105]	✓	✓	✓	Reduce the cloud load, minimise the average distance and delay for the users to get their requested applications, and improve the efficiency of the edge/fog nodes resources
[118]	✓	✓	-	Maximising the long-term accumulated reward under the constraint of computation resources
[162]	✓	✓	-	Minimise the social cost of all network service providers
[192]	✓	✓	✓	Minimising the service response time and the outsourcing traffic
[157]	✓	✓	✓	Social cost minimisation
[175]	✓	✓	-	Minimise the task processing time and long-term energy utilisation

TABLE 10. Design objectives and covered sub-problems (Continuation of Table 9).

Work	Service caching	Resource allocation	Request forwarding	Design objective
[176]	✓	✓	-	Reduce system delay while ensuring low energy consumption
[210]	✓	✓	✓	Maximising the system utility
[179]	✓	✓	✓	High availability
[158]	✓	-	-	Maximises the benefits of a service provider
[159]	✓	✓	✓	Optimisation of deadline violation, operational cost, and migration cost
[193]	✓	✓	-	Minimise the average response time
[153]	✓	✓	-	Minimise the long-term average response delay
[119]	✓	✓	✓	Maximise the total profit of all service providers
[46]	✓	✓	-	Predict the user demands based on small samples of hidden features
[82]	✓	-	✓	Minimise the long-term time average service delay under the long-term cost budget constraints
[96]	✓	✓	-	Minimising the total end-to-end latency from all users
[154]	✓	-	-	Optimise the cost of renting edge storage resources by partial caching
[163]	✓	✓	-	Maximise BS's expected profit while minimising individual service delay and cost of users
[165]	✓	-	-	Cost optimisation
[45]	✓	✓	✓	Minimise the load of the centralised cloud
[200]	✓	-	-	Minimise the total cost (activation, placement, proximity, and Co-location costs)
[202]	✓	-	-	Reduce the user-experienced tail latency, load balancing the service replicas
[203]	✓	-	-	Minimise the long-term system cost while ensuring the system stability
[198]	✓	✓	-	Maximise the time-average system throughput while stabilising the network queues

an insufficient utilisation of resources [24], [237], especially when nearby edge servers are under-load or two nearby edge servers have cached similar services. The inefficiency illuminates in cases with a high diversity of services [13].

In distributed placement scheme, cooperative placement addresses the mentioned inefficiency by considering the status of adjacent edge servers such as available resources, network status, and service demands, and the cached services to assist placement decisions of the neighbouring edge servers. In other words, edge servers share their individual local information to build a neighbourhood local view and use the view in their placement decisions. They may also collaboratively decide which services to cache and in which edge server. On the contrary, an edge server in an uncooperative distributed placement scheme uses only an individual local view such as its available resources, the cached services, and service demands for placement decisions.

4) MANAGING ENTITY

The surveyed works used a wide range of entities to manage the service placement. We categorise these entities into three main groups. It is worth noting that in hybrid methods, a combination of these entities can be used.

- Central entity: The first group includes the entities that have control over the whole network. Most of the works used a central entity to manage dynamic service placement, however due to the diversity in their system models and network architectures this responsibility in assigned to ASP [115], [116], [151], BBU [85], central cloudlet manager [201], central controller [93], central MEC server [114], cloud [4], [14], [22], [113], [169], edge orchestrator [167], [190], micro base station [117], network operator [204], service cache controller [81], and SDN controller [85], [87].
- Edge server: The second group consists of entities that can only cover part of the network. Some of these entities are (MEC-enabled) base station [13], [46], [84], [105], [118], [162], [168], [171], [172], control node [187], edge cloud [10], [137], [206], [208], edge servers [3], [83], [194], and in-network computing provider [160].
- User device: The controlling scope of the last group [177], [189], [205] is a single network node in terms of offloading tasks/services.

5) CACHING TYPE

The service caching policy can be reactive or proactive. The service caching decision in the reactive case is made after the service has been requested [24]. However, the caching decision in the proactive case is based on the perdition of request patterns. The prediction can be based on service popularity estimations, or it can be according to the prediction of demands for the next time window that are calculated based on the demands in the current window [170].

6) CACHING DECISION

The deterministic schemes employ methods that will return the same decision about which service has to be cached given a particular input parameter. Nonetheless, the optimal cache placement cannot consistently implement without errors [24]. On the other hand, probabilistic schemes use some degree of randomness by using random distributions to capture the uncertainty regarding the network dynamics and spatio-temporal demand patterns.

7) OPTIMIZATION TECHNIQUES

In the literature, a wide range of techniques are employed for dynamic service placement methods. The methods used these techniques to model the problem as well as introducing novel solutions. Tables 11 and 12 present the optimisation techniques for each work. It also shows that several works combined different techniques. In the following, we review these techniques.

TABLE 11. Taxonomy of service placement solutions.

Work	Service instance type	Distributed/ Centralised	Cooperative/ Uncooperative	Managing entity	Proactive/ Reactive	Probabilistic/ Deterministic	Optimisation techniques
[85]	-	Centralised	N/A	BBU / SDN controller	Reactive	Probabilistic	Mixed-integer linear programming, approximation algorithm (randomised rounding), and hill-climbing
[83]	-	Centralised	N/A	Central entity	Reactive	Probabilistic	Game theory (three sided cyclic game)
[83]	-	Distributed	Cooperative	Edge server	Reactive	Probabilistic	Game theory (three sided cyclic game)
[178]	Containers	Distributed	Uncooperative	Intermediate nodes	Proactive	Deterministic	Popularity-based
[123]	Lightweight virtualisation	Centralised	N/A	Central entity	Reactive	Probabilistic	Greedy algorithm
[114]	-	Centralised	N/A	Central MEC server	Reactive	Deterministic	Utility maximisation
[150]	-	Distributed	Uncooperative	MEC server	Reactive	Deterministic	Auction-based (forward auction)
[205]	-	Distributed	Uncooperative	Mobile device	Reactive	Probabilistic	Contextual multi-armed bandit and Thompson-sampling based online learning
[177]	-	Distributed	Cooperative	User device agent	Reactive	Deterministic	Greedy heuristic
[86]	-	Centralised	N/A	SDN controller	Reactive	Deterministic	Greedy heuristic
[10]	-	Distributed	Uncooperative	Edge cloud	Reactive	Deterministic	Retrospective download with least recently used
[115]	VM/Container	Centralised	N/A	ASP	Proactive	Probabilistic	Contextual combinatorial multi-armed bandit
[13]	VM/Container	Decentralised	Cooperative	base station	Reactive	Probabilistic	Graph colouring, parallel Gibbs sampling, and coalitional game
[168]	-	Decentralised	Collaborative	base station	Reactive	Probabilistic	Matching theory
[93]	Functions	Centralised	N/A	Central controller	Proactive	Probabilistic	Matching game (matching theory)
[14]	-	Centralised	N/A	Cloud	Proactive	Probabilistic	Greedy heuristic and Markov process for prediction of load distribution
[15]	-	Centralised	N/A	Central entity	Reactive	Probabilistic	Mixed-integer linear programming and heuristic search
[117]	-	Centralised	N/A	Micro base station	Proactive	Probabilistic	Markov decision process and deep reinforcement learning
[183]	-	Distributed	Uncooperative	Servers	Reactive	Probabilistic	Heuristic algorithm (popularity and centrality)
[124]	VM	Centralised	N/A	Central entity	Proactive	Probabilistic	Markov decision process
[113]	-	Centralised	N/A	Cloud	Reactive	Probabilistic	Greedy heuristic and linear regression
[22]	-	Centralised	N/A	Cloud	Reactive	Deterministic	Lyapunov optimisation
[169]	-	Centralised	N/A	Cloud controller	Proactive	Probabilistic	Approximation algorithm and greedy heuristic
[170]	-	Centralised	N/A	Central entity	Proactive	Probabilistic	Model predictive control and graph theory (minimal cut)
[151]	VM	Centralised	N/A	Service provider	Proactive	Probabilistic	Model predictive control and game theory
[160]	VM	Distributed	Uncooperative	In-network computing provider	Proactive	Probabilistic	Vickrey-English-Dutch auction
[94]	Virtual function	Centralised	N/A	Central entity	Reactive	Probabilistic	Branch-and-bound and simulated annealing
[180]	-	Centralised	N/A	Central entity	Proactive	Probabilistic	Convex optimisation (semi-definite relaxation)
[152]	Functions	Centralised	N/A	Central entity	Reactive	Deterministic	Mixed-integer programs and greedy heuristics
[87]	VM/Container	Centralised	N/A	SDN controller	Reactive	Probabilistic	Lyapunov optimisation and Markov approximation
[87]	VM/Container	Hybrid	Uncooperative	SDN controller, mobile users	Reactive	Deterministic	Lyapunov optimisation and greedy algorithm
[184]	-	Centralised	N/A	Central entity	Reactive	Deterministic	Integer linear programming and greedy heuristic
[185]	-	Centralised	N/A	Network controller	Reactive	Probabilistic	Game theory
[171]	-	Distributed	Cooperative	base station	Proactive	Deterministic	Lyapunov optimisation and Gibbs sampling
[172]	-	Distributed	Uncooperative	base station	Reactive	Probabilistic	Approximation algorithm (randomised rounding)
[201]	-	Centralised	N/A	Central cloudlet manager	Reactive	Deterministic	Greedy heuristic
[206]	-	Hybrid	Cooperative	Central entity, Edge cloud	Proactive	Deterministic	Neural network and data mining
[137]	-	Distributed	Uncoordinated	Edge cloud	Reactive	Deterministic	Mixed-integer linear programming and ranking policy
[186]	-	Centralised	N/A	Central entity	Reactive	Deterministic	Heuristic algorithm
[204]	-	Centralised	N/A	Network operator	Proactive	Deterministic	Lyapunov optimisation
[208]	-	Hybrid	Uncooperative	Edge cloud, Central entity	Reactive	Deterministic	Greedy algorithm (knapsack problem) and federated learning
[181]	Container	Centralised	N/A	Central entity	Reactive	Deterministic	Genetic algorithm, heuristic algorithm, and Artificial Bee Colony algorithm
[194]	VM/Container	Distributed	Uncooperative	Edge server	Reactive	Deterministic	Retrospective download with least-requested deletion
[81]	VM	Centralised	N/A	Service cache controller	Reactive	Deterministic	Greedy heuristic
[23]	-	Centralised	N/A	Central entity	Reactive	Deterministic	Lyapunov optimisation
[199]	Container	Centralised	N/A	Central entity	Reactive	Deterministic	Graph theory minimal cuts (max-flow algorithm)
[3]	-	Distributed	Uncooperative	Edge server	Reactive	Deterministic	Heuristic priority-based according to delay-sensitivity
[155]	-	Centralised	N/A	Central entity	Reactive	Probabilistic	Mixed-integer linear programming and greedy heuristic with probabilistic shadow request scheduling
[4]	-	Centralised	N/A	Cloud	Proactive	Deterministic	Integer linear programming
[161]	VM	Centralised	N/A	Central entity	Reactive	Deterministic	Branch and bound method and interior point method
[116]	Container	Centralised	N/A	ASP	Proactive	Probabilistic	Contextual combinatorial multi-armed bandit
[187]	-	Decentralised	Uncooperative	Control node	Proactive	Deterministic	Integer linear Programming
[166]	VM/Container	Centralised	N/A	Central entity	Proactive	Probabilistic	Mixed-integer linear programming
[167]	VM	Centralised	N/A	Edge orchestrator	Reactive	Deterministic	Fuzzy logic
[188]	VM	Centralised	N/A	Central entity	Proactive	Deterministic	Graph theory heuristic
[189]	-	Distributed	Uncooperative	User device	Proactive	Probabilistic	Game theory (strategic game)
[84]	-	Distributed	Cooperative	Small base station	Proactive	Probabilistic	Game theory (coalitional game)
[190]	-	Centralised	N/A	Edge orchestrator	Reactive	Deterministic	Mixed-integer nonlinear programming and convex optimisation
[95]	VM	Centralised	N/A	Central entity	Proactive	Deterministic	Graph theory (minimum-weight maximum matching) and autoregression prediction
[195]	-	Centralised	N/A	Central entity	Reactive	Deterministic	Graph theory (weighted maximum matching)
[173]	-	Hybrid	Uncooperative	Centralised controller, Cloudlet	Proactive	Deterministic	Matching game and task popularity
[207]	VM/Container	Centralised	N/A	Central entity	Reactive	Deterministic	Scoring algorithm
[209]	-	Centralised	N/A	Central entity	Proactive	Deterministic	Mixed-Integer Linear Programming, Heuristics (matching and local search)
[174]	Container/Function	Decentralised	Cooperative	Community leader	Proactive	Deterministic	Mixed-integer programming, Control theory
[164]	-	Distributed	Uncooperative	Cloud service operator	Reactive	Deterministic	Stackelberg game
[156]	-	Centralised	N/A	Central entity	Proactive	Deterministic	Genetic algorithm
[191]	Container	Centralised	N/A	Central controller	Proactive	Probabilistic	Integer linear programming, Bayesian Optimisation based iterative reinforcement learning
[182]	VM	Centralised	N/A	Central entity	Proactive	Deterministic	Mixed-integer linear programs
[196]	-	Distributed	Uncooperative	Base station	Proactive	Probabilistic	Markov decision process, deep Q network method
[197]	-	Distributed	Uncooperative	Base station	Proactive	Probabilistic	Markov decision process, deep reinforcement learning algorithm
[105]	Container	Distributed	Uncooperative	ICN-node	Reactive	Deterministic	Heuristic
[118]	-	Distributed	Uncooperative	Base station	Proactive	Probabilistic	Multiple choice knapsack, combinatorial multi-armed bandit
[162]	VM	Distributed	Cooperative	Network service provider	Proactive	Probabilistic	Integer linear program, randomised rounding algorithm, coalition formation game
[192]	-	Distributed	Cooperative	Edge server	Proactive	Probabilistic	Mixed-integer nonlinear programming, Gibbs sampling, Heuristic algorithm based on water filling
[157]	-	Distributed	Cooperative	Front-end server	Proactive	Deterministic	Lyapunov optimisation, reverse auction
[175]	-	Centralised	N/A	Pool network	Proactive	Probabilistic	Deep reinforcement learning

TABLE 12. Taxonomy of service placement solutions (Continuation of Table 11).

Work	Service instance type	Distributed/ Centralised	Cooperative/ Uncooperative	Managing entity	Proactive/ Reactive	Probabilistic/ Deterministic	Optimisation techniques
[176]	-	Distributed	Uncooperative	Access point	Proactive	Probabilistic	Mixed-integer nonlinear programming, Lyapunov optimisation, and Gibbs sampling
[210]	VM	Distributed	Cooperative	Edge server	Proactive	Probabilistic	Lyapunov optimisation, distributed Markov approximation algorithm
[179]	Container	Distributed	Cooperative	Edge servers	Proactive	Probabilistic	Monte Carlo Sampling, genetic algorithm
[158]	VM	Centralised	N/A	Central entity	Proactive	Probabilistic	Heuristic (Markov process)
[159]	VM/container	Centralised	N/A	Infrastructure provider	Proactive	Probabilistic	Genetic algorithm
[193]	-	Centralised	N/A	Central entity	Proactive	Deterministic	Dynamic programming
[153]	-	Centralised	N/A	Cloud centre	Proactive	Deterministic	Lyapunov optimisation
[119]	-	Centralised	N/A	Central entity	Proactive	Deterministic	Lyapunov optimisation
[46]	VM/container	Centralised	N/A	Central entity	Proactive	Probabilistic	Integer Linear Program, multi-armed bandit, generative-adversarial network (GAN)
[82]	VM/container	Centralised	N/A	Network operator	Proactive	Deterministic	Lyapunov optimisation
[96]	Functions	Centralised	N/A	Central entity	Proactive	Deterministic	Neural network, Integer linear programming
[154]	-	Centralised	N/A	Central entity	Proactive	Probabilistic	Markov Decision Process
[163]	VM	Centralised	N/A	Base station	Proactive	Probabilistic	Game theory (Stackelberg game)
[165]	-	Centralised	N/A	Edge server	Proactive	Deterministic	Heuristic
[45]	-	Centralised	N/A	Central entity	Proactive	Probabilistic	Randomised rounding technique
[200]	Container	Centralised	N/A	Central entity	Reactive	Deterministic	Graph theory minimal cuts (max-flow algorithm), Optimal stopping theory
[202]	Container	Centralised	N/A	Central entity	Probabilistic	Proactive	Random search heuristic
[203]	Container	Centralised	N/A	Central entity	Deterministic	Reactive	Lyapunov optimisation
[198]	Container	Centralised	N/A	Edge server	Deterministic	Proactive	Lyapunov optimisation

a: APPROXIMATION ALGORITHM

Approximation algorithms are efficient algorithms that find near-optimal (approximate) solutions to NP-hard optimisation problems. Wang *et al.* [169] propose an approximation algorithm with polynomial time-complexity to find the service placement online and in real-time. Ouyang *et al.* [87] design an approximation algorithm based on Markov approximation to obtain a near-optimal solution for real-time service placement optimisation. Ning *et al.* [210] apply distributed Markov approximation algorithm to determine the service placement configurations. Poularakis *et al.* [172] use an approximation approach (randomised rounding technique) that achieves provably close-to-optimal performance. Yin *et al.* [85] use a randomised rounding technique to generate multiple promising solution candidates as initial feasible ones, further improved by the hill-climbing strategy to find a near-optimal solution with high probability.

b: AUCTION-BASED

This class of techniques are used in buyer/seller or supply/demand situations to regulate the monetary interactions. Luo *et al.* [150] adapt the forward auction that deals with multiple buyers (smart devices) and a single seller, which offers the computational resource. To perform robust resource allocation, Tasiopoulos *et al.* [160] utilise the Vickrey-English-Dutch auction applied where the demand/supply conditions evolve over time and develop the unique minimum, competitive equilibrium prices. Li *et al.* [157] provide a cooperative method that encourages edge clouds to truthfully disclose workload and resource cost by utilising a reverse auction mechanism.

c: CONVEX OPTIMIZATION

Liu *et al.* [190] use convex optimisation theory to solve the multi-object optimisation problem of minimising the overall service latency while maximising the total analytics accuracy of the mobile AR users. Xing *et al.* [180] use the semi-definite programming method (semi-definite relaxation) to solve joint computation offloading and caching optimisation problem.

TABLE 13. Considered resource types (P:Processing, S:Storage, M:Memory, B:Bandwidth, E:Energy).

Work	P	S	M	B	E
[83, 152, 207, 191, 159, 193, 96]	✓	✓	✓	✓	-
[151]	✓	✓	✓	-	✓
[185]	✓	✓	-	✓	✓
[137, 81, 187, 156, 105, 118]	✓	✓	✓	-	-
[85, 183, 94, 184, 172, 208, 155, 161, 116, 188, 196, 192, 157, 210, 119, 45]	✓	✓	-	✓	-
[171, 186, 176]	✓	✓	-	-	✓
[15, 117, 124, 180, 189, 84, 175]	✓	-	-	✓	✓
[174, 182]	✓	-	✓	✓	-
[14, 238, 197, 163]	✓	✓	-	-	-
[114, 168, 93, 113, 169, 201, 23, 167, 190, 95, 209, 162, 46]	✓	-	-	✓	-
[177, 3]	✓	-	-	-	✓
[195]	-	-	-	✓	✓
[178, 150, 205, 86, 115, 13, 22, 170, 87]	✓	-	-	-	-
[10, 194, 153]	-	✓	-	-	-

d: DATA MINING

Frequent pattern mining is a data mining technique used in [206] to discover frequent movement patterns of mobile users by tracking their location, speed, and directions.

e: FUZZY LOGIC

This method is mainly used to cope with the rapidly changing uncertain systems. Low computation complexity is one of the advantages of this method, essential in online and real-time problems. Sonmez *et al.* [167] use this method to solve the workload orchestration problem in edge computing systems.

f: GAME THEORY

This theory provides powerful tools for designing distributed mechanisms by offering a wide range of games for

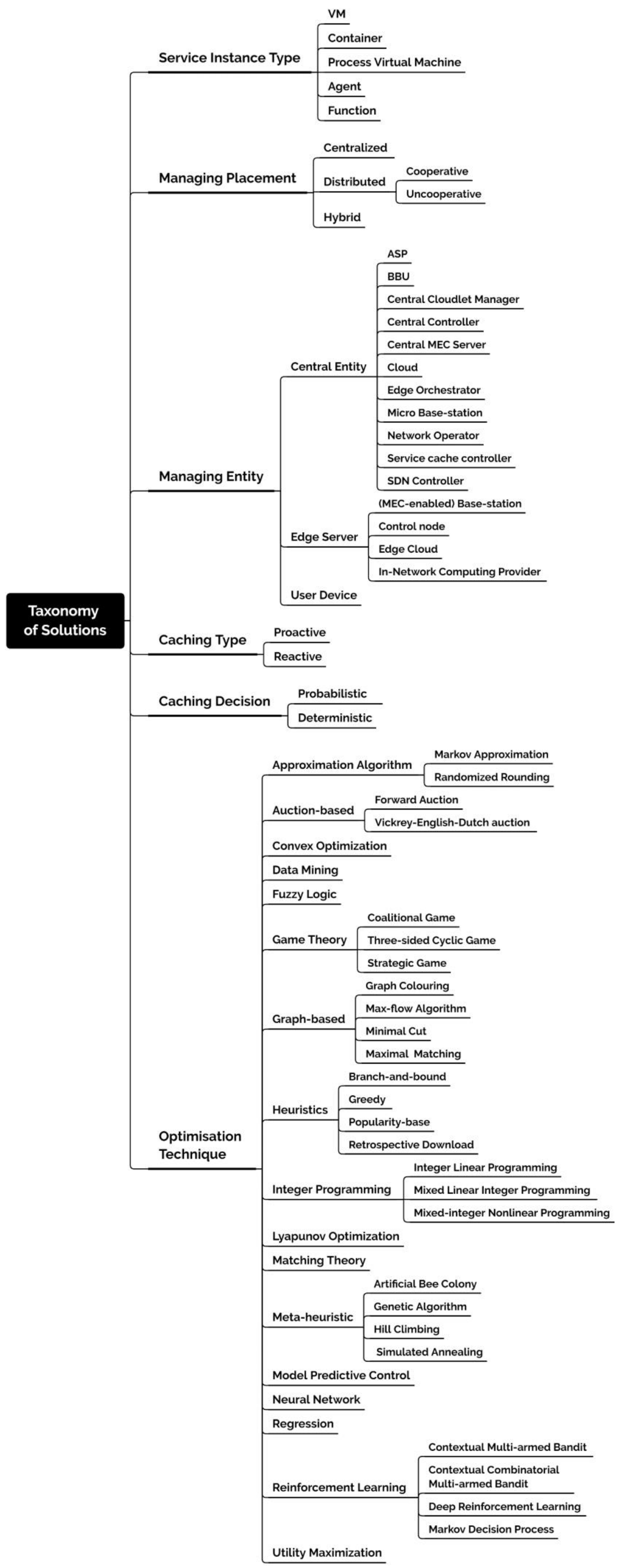

FIGURE 13. **Taxonomy of dynamic service placement solutions.**

different circumstances. Several works used game-theoretic approaches. Li *et al.* [185] use a sequential game to formulate the offloading decision. Zhang *et al.* [151] apply a multi-person non-cooperative game where multiple service providers compete for revenue to study the selfish manner of service providers. Chen *et al.* [84] apply the coalitional game to study small base station (SBS) coalition formation and the associated workload offloading decisions. Later, the authors in [13] employ the coalitional game to understand the strategic selfish behaviour of base stations that may be reluctant to participate in the collaborative edge system and then investigated how to incentivised them to form coalitions. Xu *et al.* [162] apply a coalition formation game to minimise the social cost of all network service providers. Ma *et al.* [83] introduce a three-sided cyclic game (3CG) involving users (to select preferred services), edge nodes (select high-value users), and service providers (to select cost-effective edge nodes). Chen *et al.* [189] propose a strategic game for achieving efficient computation offloading decision making among multiple mobile device users for mobile-edge cloud computing. Liu *et al.* [164] apply the Stackelberg game to formulate the interactions among cloud service operator and edge server owners to maximise the utilities of cloud service operator and edge server owners. Yan *et al.* [163] also employ the Stackelberg game of incomplete information for managing service caching and task offloading.

g: GRAPH-BASED

Several works used graph theory-based methods, including graph colouring, minimal cut, and maximal matching. Chen *et al.* [13] introduce a method based on parallel Gibbs sampling that uses graph colouring to optimise service placement decisions. Zhang *et al.* [170] use graph-theoretic minimal cuts to contribute to dynamic service placement problem for VR group gaming. Wang *et al.* [199], [200] formulate a combinatorial optimisation problem and construct a graph to encode all the costs. Then, they convert the cost optimisation into a graph cut problem and use max-flow algorithms to solve minimal cut. Jia *et al.* [95] address maximising the number of admitted service requests, while minimising their admission cost by reducing the problem to a series of minimum weight maximum matching in auxiliary bipartite graphs. Xia *et al.* [195] reduce the task offloading problem (decision about local execution, offloading to the local cloudlet, or offloading to the remote cloud) to a weighted maximum matching problem in a bipartite graph to solve it.

h: HEURISTICS

In optimisation problems, a heuristic is a technique that aims at solving a problem more quickly than traditional methods by trading completeness, accuracy, being optimum, or precision for speed. Some work use heuristic methods in their proposed approaches namely greedy heuristics [14], [81], [86], [87], [113], [123], [152], [155], [169], [177], [184], [201], [208], popularity-based [173], [178], [183], retrospective download [10], [194], and branch and bound [94].

i: INTEGER PROGRAMMING
Integer programming is a mathematical optimisation program in which some or all of the variables are restricted to be integers. Various forms of this technique are applied including Integer linear programming [4], [46], [96], [162], [184], [187], [191], mixed-integer linear programming [15], [85], [137], [155], [166], [174], [182], and mixed-integer nonlinear programming [176], [190], [192].

j: LYAPUNOV OPTIMIZATION
Lyapunov is a classical stochastic optimisation framework which splits dependent decisions (asymptotically optimally, as opposed to optimally) into deterministic optimisation problems solved at every time slot [198]. It is extensively used in control theory to ensure different stability forms of a dynamic system. These work [22], [23], [46], [82], [87], [157], [171], [176], [198], [200], [204], [210] formulate their problem as a Lyapunov optimisation.

k: MATCHING THEORY
Matching theory is a mathematical framework that optimally matches the elements of two distinct sets according to their individual preferences. It is used in combinatorial problems in which members of one set are interested in forming matching pairs with the other set. Yu *et al.* [168] transform the service placement problem into a matching problem to optimise service placement decisions and the association between BS and user device. Chiti *et al.* [93] formulated the virtual function placement problem as a matching game with externalities between virtual functions and hosting nodes. Elbamby *et al.* [173] apply a matching game between user nodes and cloudlets where the user requests are matched to serving cloudlets.

l: META-HEURISTICS
This class of approaches can potentially provide a sufficiently good solution to problems with large solution spaces that cannot be explored thoroughly. Two naturally-inspired methods (genetic algorithms and artificial bee colony) are used in [181] to optimise re-deployment solutions in microservice system. Zhao *et al.* [179] apply a genetic-based method to obtain the asymptotically optimal solution for redundant service placement problem. Maia *et al.* [156] introduce a genetic algorithm to solve their multi-objective placement problem that emphasises latency-sensitive applications. They also use a genetic algorithm to distribute the load of an application placed in different locations [159]. Simulated annealing is another meta-heuristics approach used in the literature. Bhamare *et al.* [94] apply simulated annealing as a probabilistic technique to approximate the global minimum solution which is accepted if all the latency and capacity constraints are within the acceptable range with probability one. The hill-climbing technique is another metaheuristic method used in [85] to improve initial promising candidate solutions developed by the randomised rounding technique.

m: MODEL PREDICTIVE CONTROL
Model predictive control framework is used for solving online control problems. Zhang *et al.* [151] introduce a model predictive control based method to provide an online adaptive control mechanism to reduce service provider costs, including resource allocation and reconfiguration costs for dynamic service placement. Zhang *et al.* [170] investigate provisioning problem of services that face prohibitive bandwidth and the stringent delay requirements by using a model predictive control framework to construct an online predictions algorithm.

n: NEURAL NETWORK
For online prediction of the most popular services, Wei *et al.* [206] apply a neural network model in which the training process is based on the back-propagation algorithm. The prediction model is further used in a service cache selection algorithm.

o: REGRESSION
Linear regression is used in [113] to predict the number of user devices for all service types in a future time slot. The online prediction is combined with the service caching strategy to dynamically make caching decisions.

p: REINFORCEMENT LEARNING
Multiple works formulate their research problems as a multi-armed bandit (MAB) problem and use reinforcement learning methods. Ouyang *et al.* [205] formulate the dynamic service placement problem as a contextual MAB problem to cope with the unavailable future system information and unknown system dynamics and then use Thompson-sampling based online learning method to address the problem. Chen *et al.* [116] formulate combinatorial contextual bandit learning problem to address the unknown and fluctuating service demand among changing user populations and make optimal spatial-temporal dynamic edge service placement decisions. In another work [115], they apply contextual combinatorial multi-armed bandit to study the problem of edge resource rental to an ASP in which the ASP learns service demand patterns for individual edge servers. Some of the works model their research problem as a Markov decision process and use reinforcement learning techniques. Chen *et al.* [117] model the data-intensive application edge deployment problem as a Markov decision process and use a deep reinforcement learning strategy to formulate the optimal policy to maximise the long-term discount reward. Plachy *et al.* [124] use reward function from the Markov decision process to predict user mobility and design a cooperative algorithm for dynamic VM placement. Nath *et al.* [191] introduce an orchestration framework that uses Bayesian Optimisation-based iterative reinforcement learning algorithm to find out a micro-service allocation according to the current workload. Hao *et al.* [197] propose a deep reinforcement learning method to maximise long-term average

delay reduction. Lan *et al.* [175] devise a deep reinforcement learning-based method to minimise task processing time and long-term energy utilisation.

q: UTILITY MAXIMIZATION

Samanta *et al.* [114] model adaptive service offloading as a utility maximisation problem to maximise the revenue (minimise price) and service utilisation (minimise the service latency) by considering the service priority of edge devices.

E. INSIGHTS

The literature review on dynamic service placement methods reveals that a minority embraces particular service packaging. From those, a small portion measures the associate costs of transferring service instances regarding instances' downloading time, required bandwidth, and launch time. This oversimplification raises concerns over the practicality of these methods. Our survey also shows that most methods assume edge server heterogeneity and service heterogeneity from a research directions viewpoint. However, privacy/security issues and service inter-operation do not receive appropriate attention from the researchers. The trending micro-service architecture and the growing concerns over GDPR magnify the importance of these research directions. Our analysis of design objectives also indicates that improving service-level QoS is the most popular objective, and most researchers tend to introduce some other objectives alongside QoS improvement. However, it is essential to investigate the method's overheads on various resources (e.g., processing, bandwidth, energy) and minimise them as side objectives. From the resource management perspective, we learn that the processing resource is the most important, while energy and memory are the least focused. The movement towards reducing carbon footprint by cloud infrastructure and edge servers motivates investigating the energy efficiency of the dynamic service placement methods.

VII. APPLICATION LAYER PERSPECTIVE

This section focuses on the application layer perspective of the dynamic service placement problem.

QoS is the primary concern in dynamic service placement methods from an application viewpoint, including QoS levels and factors. Application QoS can typically be categorised into three levels [239]. The first level is guaranteed services (hard QoS) that have strict hard real-time QoS guarantees. This level is suitable for safety-critical applications such as remote surgery. The second level is soft QoS that does not require hard real-time guarantees but needs to reconfigure and replace failed services. Finally, the last level is the best effort, where there are no guarantees when a service fails.

According to the surveyed papers, time-related QoS factors receive much attention compared to others. Some of these factors, such as application response time and user-perceived latency, are applied to both soft QoS and hard QoS. Other factors such as the worst application completion time and the number of applications in outage focus on hard QoS. The next group of QoS factors concentrates on throughput and resource utilisation, namely processing, network, and energy resources. It shows how effectively the edge nodes are being used and that the load is being spread evenly across them, and no one edge node is overloaded. Modern applications, such as augmented reality and autonomous vehicles, have massive network throughput. Energy efficiency is a concern to both users and edge infrastructure providers. Security is another QoS factor that is addressed in a few work. With the new legislation, such as GDPR, privacy concerns are becoming as essential as other security factors when developing a service placement method.

MEC enables the processing of exabytes of data near where it is required and generated. Such proximity benefits applications from different domains as they can address challenges regarding data volume, interoperability, and latency [240]. In the following, we review these application domains that can benefit from dynamic service placement mechanisms to address these challenges effectively and efficiently use the available resources in MEC architectures.

A. ONLINE GAMING

Cloud gaming applications provide mobile users with the required computation resources for offloading the game running and rendering then play the game video sent back [20]. However, the latency requirement plays a critical role in QoE, where the latency of more than 30–50 msec will result in degradation of gaming quality [77], especially in first-person shooter games. The MEC environment can address this limitation by designing the workflow of gaming and placing tasks onto suitable infrastructures [20].

B. AUGMENTED/VIRTUAL REALITY

The AR, VR, and their combination, known as mixed reality (MR), are a new way of interacting with the virtual world that can be applied in various use cases, including telepresence, tourism industry, smart transportation networks, and robotic-assisted surgeries [72]. The services which are employing these technologies can benefit from dynamic service placement to facilitate interactive features and fulfil their stringent latency requirement. Dynamic service placement must deal with different requirements and objectives when orchestrating services and managing resources in such applications. Liu *et al.* [190] propose an edge network orchestrator to enable fast and accurate object analytics at the edge for mobile augmented reality (MAR). This orchestrator decides when to offload computation tasks to edge servers according to their proximity using a server assignment and frame resolution (FACT) algorithm. This algorithm looks for the decision that optimises network latency, computation latency and objects analysis accuracy. Wang *et al.* [199], [200] consider social VR applications and investigate the problem of placing service entities. The applications consist of two parts: service entity (SE) and client entity (CE). The CE is placed on a user device and is responsible for monitoring user behaviours and displaying the video frames rendered by the SE. The

SE is the user's personal data bundle and takes care of the user state and computation-intensive tasks. The placement of service entities is challenging because the edge servers have different activation and running costs. Moreover, whenever users update their scenes and interacting with each other in social VR applications, SEs have to exchange metadata with the associated CEs and other SEs. The authors devise a solution based on the $s-t$ cut problem in graph theory. Zhang *et al.* [170] study mobile VR group gaming services and address dynamic rendering-module placement problem in a distributed rendering architecture where individual rendering module (IRM) renders player-specific foregrounds. The common background environments are rendered by common rendering module (CRM). According to the system architecture, each player runs an IRM on the mobile VR device. However, the placement of CRM modules is challenging due to multiple intertwined conflicting system objectives, including optimising the service's operational cost and the users' end-to-end performance. The authors model this problem as the minimal $s-t$ cut problem in graph theory and propose an online placement problem that uses the model predictive control (MPC) technique. Zhang *et al.* [216] address how to distribute the work among the user device, edge servers, and the cloud infrastructure in massively multiplayer online games (MMOGs) that use VR technologies. They apply the Markov decision process in the devised service placement algorithm dynamically places a user's gaming service on edge servers while the user moves through different access points.

C. VIDEO ANALYTICS

Real-time video analytics is widely used in traffic cameras, surveillance cameras, smartphones, in-vehicle cameras, and drone cameras. These applications are compute-intensive, and real-time processing is challenging for resource-constrained edge devices [207]. Besides, processing the videos in a cloud infrastructure requires significant bandwidth and results in a noticeable latency [184]. The placement of video analytic services in edge servers can be a solution for real-time video analytic [168]. Aral *et al.* [207] use a video streaming scenario to study the effect of different deployment solutions, such as Cloud, CDN, and Edge, on user response time.

D. SMART CITY SERVICES

Smart cities provide services from different domains (e.g., public transport, traffic monitoring, location services, intelligent public spaces, energy-saving, public safety, and emergency services) to support citizens' daily activities [4], [241]. Such services are likely to be deployed in large, dynamic, heterogeneous, and distributed environments that require dynamic service placement to satisfy citizens' needs [42], [242], [243]. The collaboration among edge servers supports the application services that require processing a large volume of geographically distributed data [72], [244]. Location-based service can benefit from dynamic service placement to adapt to the spatio-temporal demand pattern changes. Moreover, edge servers at user proximity can offer their computational resources to reduce service latency and provide fast smartphone-based services to emergency-related departments [74]. Velasquez *et al.* [4] propose a modular architecture for IoT services placement in smart cities, which considers the dynamic network traffic. A service orchestrator estimates future network traffic based on historical network status and decides the strategy to follow. The placement module uses this strategy together with an ILP solver to minimise the traffic when placing services.

E. VEHICULAR APPLICATIONS

MEC is a technology enabler for several vehicular application use cases [40]. Dynamic service placement can address the mobility challenges of vehicular applications [87] by selecting the closest node to the user for service placement [245]. It can provide high availability for safety-critical service [179], computation resources for compute-intensive use cases (e.g., autonomous driving), and low latency communication for V2X use cases (e.g., grouping-based cooperative driving, vehicle platooning, and sharing various information, including state map, environment, and traffics).

F. HEALTHCARE

Mobile health applications have a noticeable effect on our daily life. The utilisation of low-power wearable and medical sensors requires further processing and storage for monitoring health-related data and tracking records. The edge servers can provide these processing resources, especially when ultra-low latency is required. Moreover, humanoid robots sitting next to an older person may need a quick response time to offer by using edge servers [72]. Remote surgery is another use case, especially when collaborations among surgeons present in different locations is required.

G. FACTORY AUTOMATION

Automatic and real-time control of machines and systems, known as factory automation, is a potential area that can employ both 5G and MEC to reduce latency and increase network reliability [77]. Edge servers can support lightweight industrial Internet of things (IIoT) devices by bringing computing resources close to them. Some use cases include process automation, human-machine interfaces, logistics and warehousing, and monitoring and maintenance [76].

H. TACTILE INTERNET

This is the next evolution of IoT characterised by reliability, ultra-low latency with extremely high availability, and security that combines various technologies, including 5G and MEC [76]. Some candidate tactile Internet applications that can benefit from dynamic service placement are robotics, telepresence, and teleoperation.

I. INSIGHTS

Our literature review shows that most researchers consider generic applications for their methods. However, online gaming, MR, factory automation, and vehicular applications have unique characteristics that cannot be captured in a generic scenario. We learn that scenario-based solutions such as online gaming, smart city surveillance, and autonomous driving can present a valuable context for capturing various aspects of the dynamic service placement problem.

VIII. EVALUATION METHODS

This section focuses on the evaluation metrics and methodologies considered in the literature for evaluating the dynamic service placement solutions.

A. EVALUATION METRICS

Evaluation metrics are used to report or demonstrate the relative performance of the proposed service placement technique(s). We classify these metrics into the following groups.

1) TIME-RELATED METRICS

This group of metrics concentrate on time-related factors of QoS and user experience. These metrics are time variable (reported values change over time under the same settings); hence they are usually observed repeatedly and reported using the minimum and maximum, mean, standard deviation, and percentile summaries.

- `User-Perceived Latency`: It refers to the time quantification of a delay elapsed between a service request and the response received [87], [204], [205]. Some similar metrics are also reported in the literature, including response time [153], [174], [192], [193], [206] and service latency [114], [178], [183].
- `Round-Trip Delay`: The time taken by a client request to reach the (MEC) system, getting executed and then results returned to the client [246]. Similar metrics such as an end to end latency [186] and network latency [201], [207] are also reported in the literature.
- `Execution time`: This metric shows the amount of time spent to process a computational task (e.g., fulfil a user request) [95], [114], [150], [160], [207], [209], which is also referred to as completion time [93], [94].

2) RESOURCE UTILISATION AND OVERHEADS

With the availability of limited resources, an essential objective of dynamic service placement methods in a MEC environment is to optimise resource utilisation while satisfying the QoS requirements of services. Many studies use the term cost metrics to demonstrate the efficiency of proposed technique(s) by measuring the utilisation of available hardware resources while performing user tasks while considering dynamic service placement.

- `Network utilisation`: The amount of network bandwidth utilised to perform user tasks by applying a dynamic service placement method [96]. Some work uses network throughput [148], [172] and backhaul traffic [188] to evaluate network usage.
- `CPU utilisation`: The amount of CPU used (usually measured in percentage) over time to complete user-requested services by applying a dynamic service placement method [152], [167], [171], [172], [178]. VM utilisation is another similar metric used in [167].
- `Memory utilisation`: The amount of memory (measured in MBs) utilised to fulfil user-requested service by applying a dynamic service placement method [191].
- `Energy consumption`: The energy consumption metric comprises several items, and it is used both as a performance metric and a constraint. Xing *et al.* [180] use local execution, task offloading, computation uploading, and remote execution energy consumption. Lan *et al.* [175] consider the consumed energy for device task offload, communication between RSUs, and communication between RSU and the cloud. Xu *et al.* cite129 model the energy consummation for both broadcast and point to point communication. The authors in [171] model base station's computation energy consumption.
- `Service switching cost`: This metric measures the overheads to allocate various resources and deactivate services due to the reconfiguration of services in an edge server [14], [113], [166], [170], [205].

3) THROUGHPUT

This group of metrics concentrates on the number/rate of service processing in edge servers [198].

- `The number of served/admitted service requests`: The resource-constrained edge servers can host and process a limited number of services. This metric measures the throughput of these edge servers in terms of the number of served services or the number of served users [184]. Since the response time of the services running in edge servers are shorted than the ones executing in the cloud, the goal is to maximise the number of served/admitted services in edge servers [95], [155]. This metric can show the number of fulfilled service requests considering their stringent tolerable delays (satisfaction rate [137]).
- `Rejected requests`: The metric is applied in cases where service requests have maximum tolerable delays, and they are prone to fail due to various types of delay (e.g., network delay, processing delay, queuing delay) [151]. Some of the other similar metrics are: unsatisfied service request [94], failed tasks [167], and deadline violation [156], [159].
- `Cache hit rate`: This metric calculates the percentage of service requests satisfied by the edge server over the total number of requests. The higher hit rate shows that more service requests have benefited from service caching [81], [105], [173], [193].

- `Forwarded traffic load`: The number of service requests forwarded to the cloud is a performance metric of a dynamic service placement method in which reducing the number of forwards or the forwarded traffic leads to performance improvement [113], [168].
- `Average queue length`: Edge servers maintain queue(s) of unfinished task requests, which can be used as an evaluation metric [157], [158], [203].

4) REVENUE
Application service providers rent their required resources and receive revenues from users by processing their service requests. This group of metrics aims to evaluate this aspect of dynamic service placement solutions [161].

- `Minimise monetary cost`: This metric is used to evaluate the cost of renting the edge server resources by application service providers [117], [156], [159], [169], [200]. The metric is generally used in a combination of other metrics such as time-related ones.
- `Maximise profit`: Application service providers make a profit by processing user services while covering their service placement costs. This metric evaluates the gained total profit due to dynamic service placement [46], [83], [85], [154], [160].

B. EVALUATION ENVIRONMENT
Performance evaluation of the proposed techniques is carried out using two different environment settings: simulation modelling and realistic scenarios. Some studies have also carried out evaluation using mathematical models [148]. In the following, we outline and briefly describe these environments.

1) SIMULATORS
Evaluating the service placement strategies in a simulation environment is preferred when the realistic infrastructure is not available. Researchers also opt for simulation to avoid the setup cost and the over-complicated process of setting up the dynamic infrastructure. This choice also becomes inevitable when running repeated experiments require infrastructure reconfigurations multiple times [250]. However, simulating a realistic environment is itself a complex process. In literature, several simulation frameworks are utilised by researchers for demonstrating the viability of their proposed solutions. Harnessing the use of a particular simulation framework is intensely driven by the application needs. Many generic tools and programming environments like Matlab, JAVA, and Python are also used for developing and running simulations. In the following, we describe some popular simulation tool and frameworks reported in the literature.

a: CLOUDSIM
CloudSim is a simulation toolkit for simulating cloud computing infrastructure and application provisioning environments [251]. CloudSim has a long known and reliable codebase, and the research community has broadly used it for performing simulations with standard cloud computing scenarios. However, CloudSim does not support simulation of IoT and edge environment. Nevertheless, extending the core functionalities (e.g., the components and its event-based communication mechanism among those components) offered by CloudSim, many simulation frameworks for edge (and Fog) computing are built on top of this simulator. ContainerCloudSim, implemented as an extension of CloudSim, is a simulation architecture for containerised clouds [252].

b: IFOGSIM
The iFogSim simulation toolkit enables the modelling of IoT and Fog environments [253]. The iFogSim simulator is built on top of the CloudSim simulation framework. It allows users to simulate IoT applications and measure the impact (e.g., latency, energy and resource consumption) of resource management techniques in IoT and Fog/edge environments under different scenarios. A more recent version of the simulator named iFogSim2 offers service migration and cluster formation while enabling microservice orchestration [254].

c: EDGECLOUDSIM
The EdgeCloudSim simulator is also built on top of the CloudSim framework [255]. Unlike iFogSim, EdgeCloudSim supports the device mobility model as well as realistic load generation functionality. EdgeCloudSim allows users to simulate edge computing scenarios in which multiple edge servers can run in coordination with cloud solutions through its orchestration support.

d: IOTSIM-EDGE
IoTSim-Edge is another simulation framework that extends the capabilities of CloudSim to enable modelling the behaviour of IoT and edge computing environments [250]. IoTSim-Edge provides a composite simulation environment for IoT-edge application scenarios by supporting battery-oriented energy modelling, heterogeneous IoT devices and edge communication protocols, device mobility, and resources provisioning modelling. Unlike EdgeCloudSim, IoTSim-Edge also supports application composition modelling of IoT applications.

e: OMNET++
The OMNeT++ (a well-known) simulation framework provides an extensive set of libraries that can be used to simulate and test network functions and protocols [256], [257]. Domain-specific functionality is provided by model frameworks that are developed as independent modules. Numerous extensions are available for real-time simulation, network emulation, database integration, and several other functions. Many researchers use OMNeT++ to model specific network scenarios for IoT-edge applications.

f: FOGNETSIM++
FogNetSim++, designed on the top of OMNet++, enables researchers to incorporate customised mobility models and

TABLE 14. Service placement evaluation methods & reported performance metrics.

Work	Theoretical	Simulation	Realistic/Testbeds
[83, 123, 114, 150, 205, 86, 10, 93, 14, 15, 22, 169, 170, 180, 152, 87, 166, 164, 118, 210, 82, 163, 165, 45]	✓	✓	-
[115, 179]	✓	-	✓
[191, 202]	-	✓	✓
[85, 178, 177, 148, 13, 168, 117, 183, 124, 113, 151, 160, 94, 246, 184, 185, 171, 172, 201, 206, 137, 186, 204, 208, 194, 81, 23, 199, 3],[155, 161, 247, 116, 187, 167, 188, 189, 84, 190, 95, 195, 173, 207, 209, 174, 156, 196, 197, 162, 192],[157, 175, 176, 158, 159, 153, 119, 46, 96, 154, 200, 203, 198]	-	✓	-
[248, 249, 181, 182, 193]	-	-	✓

Fog node scheduling algorithms and manage handover mechanisms [258]. FogNetSim++ allows integration of OMNet++ modules. It allows the modelling of network characteristics (e.g., error rate and data rate), device mobility modelling, Fog node scheduling algorithms and management of task handover mechanisms.

Other simulators that allow modelling the Fog infrastructure [259], [260] and test dynamic service placement in mobile micro-clouds [246], [247] are also proposed in the literature. While there is a wide range of simulators available for cloud computing and the network domain, few can be used to simulate Fog and edge computing scenarios. In the recent past, numerous simulation tools have been developed to simulate IoT and edge computing scenarios. Many of these tools have extended functionalities from existing cloud and network simulation frameworks.

2) TESTBEDS

Several researchers deploy a cluster of small single-board computing devices as testbeds when building large and expensive testbeds may not be an option [86]. For example, authors in [148] have carried out small testbed experiments using Raspberry Pi devices and an Intel application server with Xeon CPU. Similarly, researchers in [261] deployed a testbed using Raspberry Pi nodes and HPC nodes for evaluating their proposed service placement approach. There are also few real-world and flexible testbeds available that support the evaluation and experimentation of a wide variety of other research and innovation works. In the following, we provide a brief overview of these testbeds.

a: PLANETLAB

PlanetLab [262] is an open platform for developing, deploying, and accessing planetary-scale services. A testbed has a collection of machines (1000+ nodes) distributed worldwide (500+ locations). PlanetLab serves the research community as a testbed for overlay networks where researchers can run experiments various services. Researchers can experiment with new services and models under real-world conditions on a large scale. PlanetLab allows multiple services to run concurrently and continuously, each in its slice of PlanetLab. With hundreds of research projects hosted on PlanetLab, some studies in service placement have also utilised this testbed [263]–[265]. PlanetLab was officially shut down in May 2020 [266].

b: FIT/IOT-LAB

The FIT IoT-LAB testbed [267] is an open testbed composed of over 1500+ low-power wireless nodes available for experimenting with large-scale wireless IoT technologies. The IoT-LAB testbed is spread in 6 different sites across France. These sites feature different node and hardware capabilities, available through the single web portal and common REST interfaces.

c: GRID'5000

Grid'5000 [268] is a large-scale testbed for experiment-driven research with more focused on parallel and distributed computing. Grid'5000 provides access to 800+ compute nodes (grouped into homogeneous clusters spread in 10 sites in France) for experimental research on large future infrastructures. In 2018, Grid'5000 was merged with FIT to build the super infrastructure for large-scale experimental computer science (SILECS) [269].

d: FED4FIRE+

Federation for FIRE plus (Fed4FIRE+) [270] is a successor of the Fed4FIRE project. It is developed under the European Union's Programme Horizon 2020, offering the largest federation worldwide of next generation Internet (NGI) testbeds to support experimentally driven research on networking services and applications, such as optical networking, wireless networking, SDN, cloud computing, Fog computing, data science applications, smart cities, etc.

3) THEORETICAL ANALYSIS

Several studies, for example [14], [15], [22], [114], [123], [150], [169], [180], [205], [271], also theoretically analyse the performance of the proposed dynamic service placement algorithms. This approach is often followed by simulation experiments to demonstrate the viability of proposed solutions. Table 14 summarises the selected works for the proposed evaluation approaches.

C. REAL-WORLD DATA

Some researchers use real-world data to evaluate the performance of their proposed service placement methods. We categorise the real-world data into edge server locations, mobility traces, and service requests.

1) EDGE SERVER LOCATIONS

The locations of cell towers are used as edge server locations in [155] based on the taxicab traces [272] gathered in 2009. Several works use the locations of Starbucks shops in Beijing [123], Manhattan island [123], New York City [199], and Los Angeles [199] as edge server locations since these shops usually can achieve a decent coverage for users. Zhao *et al.* [179] use real-world data based on the geolocation information of base stations and end-users within the Melbourne CBD area contained in the EUA dataset [273]. This dataset is used in [235] too.

2) MOBILITY TRACES

Several mobility traces are used in the literature. Researchers [14] use the trajectories of the taxis and the information of users' journeys to commute the metro collected on May 21st 2014, in Shenzhen city. In [170], the anonymised data traces containing 5,000 users' signalling data logs in the city of Guangzhou provided by one of the largest telecom carriers in China collected from November 15th through 21st, 2015. A dataset of 536 taxis in 30 days is utilised in [246] that includes the time, latitude, longitude, altitude, and occupancy data [274]. Wei *et al.* [206] use the GPS trajectory dataset from Microsoft Research Asia [275] and user trajectory from call detail records (CDR), which published by Orange [276]. The taxicab traces dataset [272] is used in [155] by extracting the traces of 36 users over 520 minutes with location updates every 10 minutes. Ma *et al.* [204] employ mobile users' traces using Twidere (an open-source Android Twitter). Their dataset includes 100 users with the consecutive locations obtained from the GPS timestamp. Aral *et al.* [207] implement a mobility model based on real-world check-ins that provide the temporal and spatial information of check-in behaviours in people's daily lives, including user's ID, check-in time, check-in location, and POI information. The data is authorised by a Chinese online service provider Tencent and includes over 37 million check-in records in the dataset, generated by over 350 thousand users during the second half of 2013 [207].

3) SERVICE REQUESTS

The real-world data for service demand workload can be obtained from several sources. Chen *et al.* [115] use the service demand trace dataset (AuverGrid) collected by the grid workloads archive (GWA) [277] that includes around 400,000 task requests of 5 grids that provide computational support for e-Science. The authors assume grids corresponds to small base stations in the edge network. By assuming videos correspond services, Yang *et al.* [14] use the Youtube dataset [278] that contains 10,324 videos crawled from the Youtube website on Mar 2nd, 2007. Researchers [15], [81], [237] use a dataset from Google cluster [279] that includes two kinds of data in the log files. The first is the Job-event log files that show the services that users request, and the second is the Task-event log files that show the resources required by the services. Zhang *et al.* [151] use the Worldcup 98 dataset that contains HTTP requests for a total duration of 92 days which includes several occurrences of demand spikes. Farhadi *et al.* [155] use a wireless trace [272] containing transmission timestamps generated by five different applications from 36 wireless devices to produce user requests. Chen *et al.* [116] use the data collected in [280] that contains the context information of a total of 10,208 end-users and their demand for 23 types of mobile applications. It is a helpful dataset for studying the link between the demand for mobile applications and user context information. Cicconetti *et al.* [148] use the dataset introduces in [281] containing the Internet activity with a 10 min granularity. The data is used to set a load of each cell over time. Xu *et al.* [46] extract a sample of user information from the dataset of NYC Wi-Fi hotspot locations [282] to learn the location, time, service status, and other characteristics of each request. Narayana *et al.* [165] use trace-data obtained from a Google Cluster with requests for four types of jobs/services.

D. INSIGHTS

In the recent past, research interest in the area of edge computing has gained significant momentum. Newly developed techniques and algorithms require extensive validation before the real-world deployment. Researchers and developers are looking for realistic ways to evaluate and demonstrate the viability of service placement approaches in the edge computing environments. Use of scalable real-world infrastructure is the most effective approach undoubtedly. However, unlike the cloud, deployment of edge computing infrastructure takes place in a highly distributed manner (and has high cost), which makes it challenging to harness in the real-world for experimentation purposes. Aforementioned real-world testbeds such as PlanetLab, FIT, FED4Fire+ etc. are available which support the experimentation of wide variety of research works. However, in these settings, modification and tuning of system parameters is an over-complicated process and there are long delays before researchers get their turn for running experiments. Moreover, testbeds do not represent a scalable, heterogeneous, and distributed edge infrastructure.

Due to such constraints, a number of simulation tools/frameworks have been developed which allow researchers to simulate the edge computing environments and test their approaches with different environment configurations. Several simulation tools are presented in literature but iFogSim is used by most of researchers for simulating the edge (and Fog) computing environments. However, there are threats to validity that can emerge from simulated evaluations [283], [284]. For example, real-world conditions can affect the network conditions between edge servers, service execution time, and number of services requested at a given time. Hence, more granular model of simulation environment can reduce this validation gap. Furthermore, real-world data such as edge server locations, mobility traces, and service requests may be used when simulating a scenario. Despite the plethora of available simulation tools, they are limited in terms of their functionalities and lack proper documentations, hence, developing a granular simulation model becomes a complicated task.

IX. OPEN ISSUES AND CHALLENGES

This section introduces open issues and potential enhancements of dynamic service placement methods in the MEC environment. Multi-access edge computing needs a paradigm shift in how services are dynamically placed, deployed, and executed. As different applications will need different service requirements for quality performance, new techniques with improved capacity, intelligent traffic and steering solutions will have to be developed and integrated to meet stringent requirements. Complexity and heterogeneity in 5G also impose the requirements for network operation management. Service providers can bridge the gap between the network layer and the service-enabling stack necessary to translate and define the differentiated services into the underlying parameters for the network orchestrator. This can put service providers in a solid position to manage the challenges associated with service deployment on the run to monetise successfully. Consequently, this also brings new challenges to address and fix operators' newly developed service availability mode in the "$edge + AI + 5G$" network.

We highlight several open issues and challenges that require extra attention in future work.

A. NETWORKING

1- Future applications and services in 5G/B5G will increasingly rely on AI. Besides, intelligence at the edge is necessary to automate and control network functionalities in RANs. Such near real-time applications can lead to the development of new radio intelligent controllers (RIC). To support these stateful and session-oriented applications, 5G MEC need to rely on network architectures beyond IP. ICN is a potential candidate with its apparent benefits such as fast and efficient service delivery and improved reliability. Both academic and industrial standardisation bodies need to push forward the ICN enabled edge integration for 5G.

2- Integrating edge will surely increase the 5G network infrastructure cost. The extreme cost option will be if an edge is implemented at every base station site, with likely thousands of sites per MNO (Level 1). In contrast, edge deployment at RAN aggregation points or points of presence PoPs) will reduce edge sites to a scale of hundreds per MNO and cost significantly. However, the most cost-effective solution would be deployment at a few tens of EPC sites and extensive central office facilities, but it may result in a trade-off service availability and latency. Although there is a hype about the 1-millisecond latency targets (theoretical), latency levels in the 10-12 millisecond range are more likely. This latency requirement will eventually rely on fibre and optical switching in the transport network infrastructure.

3- Shifting from hardware to software platforms can simplify the multi-tenancy support where multiple services/applications from different use cases can be accommodated using a common SDN/NFV-based 5G infrastructure system as discussed in Sections V-B1 and V-B2. The network sharing paradigm through network slicing enables multiple VNFs to be configured on the same NFV platform but creates management problems of large network slices. It is to be noted that the dynamic resource sharing of network slice among tenants would make network resource utilisation more flexible and efficient. This calls for the development of intelligent scheduling algorithms to allocate resources among slices dynamically. Moreover, the VNFs placement within the slice, intra-slice management, and inter-slice management problems need vital efforts to realise the network slicing effectiveness in 5G networks.

B. MIDDLEWARE

1- The services have different characteristics. Some need large-size input data, some large-size output data. Some services are computationally intensive, and some are lightweight. Dynamic service placement methods have to discriminate different services and take these characteristics into account while deciding their placements, particularly in multi-objective placement scenarios.

2- Collaborative service caching is an open research challenge. It requires establishing incentive mechanisms to justify cooperation between third parties that own the edge servers and may have conflicting interests regarding monetising their cached services.

3- Most of our surveyed works model the service request patterns as stochastic processes. However, in some real-work scenarios, such as social events or disaster recovery, the service requests patterns are bursty and non-stochastic. Only a few works study such bursty user requests on dynamic service placement methods.

4- The scalability service placement methods require further study. The scalability has various dimensions, including geographical distribution of edge servers, number of services, diversity in QoS requirements, number of active users, computational complexities of placement methods, and density of overlapping base stations in a highly-populated area.

5- High availability of services in their dynamic placement is a new trend of research and can be studied from various points. The redundant placement increases services' robustness, making the system resilient to failure, essential for safety-critical services. A healing mechanism is another area where quick failure detection and recovery plays an essential role. Creating multiple service instances enhances faster response to requests and helps to balance request workload received from different users at different locations.

6- The application services evolve in time to fix bugs and cope with the changes in requirements. Besides, diverse versions of an application service can offer different QoS to meet the possible trade-offs. Most of the survey papers did not consider different versions of an application service.

7- Parallel processing of dynamic service placement methods is one of the research gaps that can address the scalability issues. The design of such parallel algorithms can benefit from various multi-core and GPU-based platforms.

8- Integrating hybrid clouds (multi-cloud) with the MEC environment is another open issue. While most of the reviewed research assumes a single cloud data centre, application services can be supported by different cloud infrastructures in a real-world scenario. A collaboration between these infrastructures and the MEC environment introduces challenges for service caching, resource allocations, and service request forwarding.

C. APPLICATION

1- In the reviewed literature, partial caching of application services has not received much attention. Application services can be fully or partially cached in an edge server. In partial caching, some parts of the application, such as specific components or functions that require data-intensive inputs, are cached, while the rest of the application, which is computationally intensive and requires less data exchange, remains in the cloud. Most of the MEC dynamic service placement methods use full caching. For instance, only the most requested language is cached in a translation service.

2- The dependency between services of an application is studied in a few dynamic service placement research. While service function chaining is well-explored in static placement methods, considering complex dependency graphs of service dependencies in a microservice architecture is an open challenge for dynamic placement.

D. SECURITY/PRIVACY/TRUST

1- 5G security is even more critical and a top concern for edge and network operators. A provider's MEC network needs to inter-network with an enterprise network to integrate 5G communication capabilities and edge services into enterprise systems. In current implementations, edge routers are utilised for communicating with enterprise networks. Network security is a significant concern for both enterprise networks and service providers. A confidential computing and networking security solution is required to ensure high privacy and security. In reference to edge caching, the usual strategies are distributed and vulnerable to potential attacks, such as DoS or rogue edge attacks. Caching nodes under attack may shut the service to users or push undesired services. The malicious or compromised devices can also transmit the undesired services via D2D communications. The lack of focus on security is a worrying concern and should be evaluated more in future papers.

2- In centralised service placement methods, edge servers have to update the central orchestrator with system information containing user information, service demands, and preferences that increase the vulnerability of privacy-sensitive data. While distributed/federated learning methods are less exposed, it is an open research question that should be considered in future approaches to ensure that user data is protected under GDPR.

3- Trust has an essential role in MEC environments. On the one hand, the MEC environment is a multi-vendor ecosystem where various administrative domains (e.g., service providers, infrastructure providers, network operators, edge operators) manage different network elements (e.g., base stations, edge servers, access points). On the other hand, these network elements host application service, and the user requested services are forwarded across this environment and executed on these network elements. It brings up the need to establish an integrated trust mechanism among the network elements and, in a broader picture, among third parties that manage base stations and edge servers.

4- Dynamic service placement methods can benefit from Blockchain technology. It can be used as a distributed trust mechanism between various stakeholders in dynamic service placement. Besides, smart contracts can enhance service automation, regulate the interactions between the cached services, and support the monetary aspects of service usages.

E. EVALUATION

1- Most of the approaches have been evaluated against generic edge services or heterogeneous applications. This is good as it allows the approach to be used in several different environments. However, given the lack of approaches that have focused on specific applications, such as drones, UAV, AR/VR, autonomous vehicles, there is an open research question of developing a specific service placement approach to handle the specific requirements of that application. When analysing the papers, we focus on evaluating the dynamic service placement approach and not on the introductory text. Many papers introduce the application type at the start of the paper, but ordinary generic services are used when the approach is evaluated.

2- There is no consensus on standardised evaluation metrics that can evaluate the performance of proposed dynamic service deployment techniques. A significant number of works introduced their customised cost metrics. Besides, the research does not use standard definitions for more popular metrics such as latency.

3- In the real world, the infrastructure has properties such as large-scale, geographical distribution, and heterogeneity of edge computational nodes, making the dynamic service placement a challenging issue in such a scenario. Although it is common to evaluate proposed work via simulations, existing simulation frameworks have significant gaps in their capability to model mobility, scalability, complexity, and specific requirements of edge computing scenarios.

X. SUMMARY

The traditional cloud computing paradigm faces several limitations, including network delays and possible congestion in the backhaul network. The limitations motivate researchers toward decentralised paradigms in processing application services. The placement of these application services plays an essential role in fulfilling QoS requirements and overall system performance. This work investigated the dynamic service placement problem in a MEC environment, following systematic literature review guidelines as a research method. We described the dynamic service placement problem, explored its comprising sub-problems, and outlined the key challenges. Then, we reviewed the research from different viewpoints. First, we explored different network architectures and the critical role of networking technologies in dynamic service placement. Second, we reviewed the literature from a middleware viewpoint, introducing the research directions, devising the taxonomy design objectives, and the taxonomy of solutions for dynamic service placement. Then, we surveyed the works from an application layer perspective and discussed several application scenarios that can benefit from dynamic service placement. Additionally, we surveyed the literature based on their evaluation metrics, evaluation environments, and the real-world data they have utilised. Finally, we presented a list of open research challenges that should be addressed in future work.

REFERENCES

[1] Cisco. *Cisco Annual Internet Report (2018–2023) White Paper*. Accessed: Jan. 4, 2022. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] Ericsson. *Mobile Data Traffic Outlook*. Accessed: Jan. 4, 2022. [Online]. Available: https://www.ericsson.com/49da93/assets/local/mobility-report/documents/2020/june2020-ericsson-mobility-report.pdf

[3] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang, "Selective offloading in mobile edge computing for the green Internet of Things," *IEEE Netw.*, vol. 32, no. 1, pp. 54–60, Jan./Feb. 2018.

[4] K. Velasquez, D. P. Abreu, M. Curado, and E. Monteiro, "Service placement for latency reduction in the Internet of Things," *Ann. Telecommun.*, vol. 72, nos. 1–2, pp. 105–115, Feb. 2017.

[5] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Apr. 2010, pp. 27–33.

[6] 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. 2009.

[7] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 45–55, Nov. 2014.

[8] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.

[9] M. Patel *et al.* (2014). Mobile-Edge Computing Introductory Technical White Paper. ETSI. Accessed: Mar. 23, 2022. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf

[10] I.-H. Hou, T. Zhao, S. Wang, and K. Chan, "Asymptotically optimal algorithm for online reconfiguration of edge-clouds," in *Proc. 17th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 291–300.

[11] P. Smet, B. Dhoedt, and P. Simoens, "Docker layer placement for on-demand provisioning of services on edge clouds," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 3, pp. 1161–1174, Sep. 2018.

[12] W. Tärneberg, A. Mehta, E. Wadbro, J. Tordsson, J. Eker, M. Kihl, and E. Elmroth, "Dynamic application placement in the mobile cloud network," *Future Gener. Comput. Syst.*, vol. 70, pp. 163–177, May 2017.

[13] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 377–390, Feb. 2021.

[14] L. Yang, J. Cao, G. Liang, and X. Han, "Cost aware service placement and load dispatching in mobile cloud systems," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1440–1452, May 2016.

[15] T. X. Tran, K. Chan, and D. Pompili, "COSTA: Cost-aware service caching and task offloading assignment in mobile-edge computing," in *Proc. 16th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2019, pp. 1–9.

[16] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–37, Oct. 2019.

[17] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131543–131558, 2019.

[18] 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.

[19] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–23, Jan. 2020.

[20] 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.

[21] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[22] J. Zhou, J. Fan, J. Wang, and J. Jia, "Dynamic service deployment for budget-constrained mobile edge computing," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 24, p. e5436, Dec. 2019.

[23] J. Zhou, J. Fan, J. Wang, and J. Jia, "Service deployment for latency sensitive applications in mobile edge computing," in *Proc. 6th Int. Conf. Adv. Cloud Big Data (CBD)*, Aug. 2018, pp. 372–377.

[24] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Commun. Surveys Tuts.*, vol. 21, pp. 2525–2553, 3rd Quart., 2019.

[25] G. Chen and D. Kotz, "A survey of context-aware mobile computing research," Dartmouth Comput. Sci., Hanover, NH, USA, Tech. Rep. TR2000-381, 2000.

[26] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[27] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, vol. 6, pp. 3962–3976, 2018.

[28] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, M. Imran, and S. Guizani, "Mobile ad hoc cloud: A survey," *Wireless Commun. Mobile Comput.*, vol. 16, no. 16, pp. 2572–2589, Nov. 2016.

[29] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. de Foy, and Y. Zhang, "Mobile edge cloud system: Architectures, challenges, and approaches," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2495–2508, Sep. 2018.

[30] 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.

[31] B. Khazael and H. T. Malazi, "Distributed coordination protocol for event data exchange in IoT monitoring applications," in *Proc. 11th Int. Conf. Inf. Knowl. Technol. (IKT)*, Dec. 2020, pp. 113–118.

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

[33] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for Internet of Things services," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 16–24, Feb. 2017.

[34] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[35] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, vol. 98, pp. 289–330, Sep. 2019.

[36] D. R. Vasconcelos, R. M. C. Andrade, V. Severino, and J. N. D. Souza, "Cloud, fog, or mist in IoT? That is the question," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–20, May 2019.

[37] M. Aazam, I. Khan, A. A. Alsaffar, and E. N. Huh, "Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved," in *Proc. 11th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST)*, Islamabad, Pakistan, Jan. 2014, pp. 414–419.

[38] *Multi-Access Edge Computing (MEC) Framework and Reference Architecture*, document ETSI GS MEC 003 V2.2.1, MECISG ETSI, ETSI GS MEC, Dec. 2020.

[39] 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.

[40] 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.

[41] S. Bi, L. Huang, and Y.-J.-A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4947–4963, Jul. 2020.

[42] C. Cabrera, G. White, A. Palade, and S. Clarke, "The right service at the right place: A service model for smart cities," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2018, pp. 1–10.

[43] H. Huang, H. Zhang, T. Guo, J. Guo, and C. He, "Reliable redundant services placement in federated micro-clouds," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2019, pp. 446–453.

[44] H. Zhao, S. Deng, Z. Liu, J. Yin, and S. Dustdar, "Distributed redundancy scheduling for microservice-based applications at the edge," *IEEE Trans. Services Comput.*, early access, Aug. 3, 2020, doi: 10.1109/TSC.2020.3013600.

[45] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in MEC networks with storage, computation, and communication constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1047–1060, Jun. 2020.

[46] Z. Xu, S. Wang, S. Liu, H. Dai, Q. Xia, W. Liang, and G. Wu, "Learning for exception: Dynamic service caching in 5G-enabled MECs with bursty user demands," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Nov. 2020, pp. 1079–1089.

[47] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.

[48] 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.

[49] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin, "A survey and taxonomy on task offloading for edge-cloud computing," *IEEE Access*, vol. 8, pp. 186080–186101, 2020.

[50] M. R. Alizadeh, V. Khajehvand, A. M. Rahmani, and E. Akbari, "Task scheduling approaches in fog computing: A systematic review," *Int. J. Commun. Syst.*, vol. 33, no. 16, p. e4583, 2020.

[51] P. Hosseinioun, M. Kheirabadi, S. R. K. Tabbakh, and R. Ghaemi, "ATask scheduling approaches in fog computing: A survey," *Trans. Emerg. Telecommun. Technol.*, vol. 33, p. e3792, Jan. 2020.

[52] Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "A survey on mobility-induced service migration in the fog, edge, and related computing paradigms," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–33, Sep. 2020.

[53] Z. Tao, Q. Xia, Z. Hao, C. Li, L. Ma, S. Yi, and Q. Li, "A survey of virtual machine management in edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1482–1499, Aug. 2019.

[54] T. L. Duc, R. G. Leiva, P. Casari, and P.-O. Östberg, "Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–39, Sep. 2020.

[55] Y. Teng, M. Liu, F. R. Yu, V. C. M. Leung, M. Song, and Y. Zhang, "Resource allocation for ultra-dense networks: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2134–2168, 3rd Quart., 2019.

[56] A. G. Sheshjavani, A. Khonsari, S. P. Shariatpanahi, M. Moradian, and A. Dadlani, "Coded caching under non-uniform content popularity distributions with multiple requests," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2020, pp. 1–6.

[57] A. S. Kakar and M. S. Rohie, "Review of probabilistic techniques used for web browsers' caching," *Eur. J. Eng. Technol. Res.*, vol. 5, no. 7, pp. 773–780, Jul. 2020.

[58] G. Wittenburg and J. Schiller, "A survey of current directions in service placement in mobile ad-hoc networks," in *Proc. 6th Annu. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2008, pp. 548–553.

[59] S. Ali, A. Mitschele-Thiel, A. Diab, and A. Rasheed, "A survey of services placement mechanisms for future mobile communication networks," in *Proc. 8th Int. Conf. Frontiers Inf. Technol. (FIT)*, 2010, p. 39.

[60] M. S. Raghavendra, P. Chawla, and A. Rana, "A survey of optimization algorithms for fog computing service placement," in *Proc. 8th Int. Conf. Rel., Infocom Technol. Optim. (Trends Future Directions) (ICRITO)*, Jun. 2020, pp. 259–262.

[61] A. Brogi, S. Forti, C. Guerrero, and I. Lera, "How to place your apps in the fog: State of the art and open challenges," *Softw., Pract. Exper.*, vol. 50, no. 5, pp. 719–740, May 2020.

[62] F. A. Salaht, F. Desprez, and A. Lebre, "An overview of service placement problem in fog and edge computing," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–35, May 2021.

[63] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Gener. Comput. Syst.*, vol. 87, pp. 278–289, Oct. 2018.

[64] 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.

[65] 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.

[66] V. Sindhu and M. Prakash, "A survey on task scheduling and resource allocation methods in fog based IoT applications, "in *Communication and Intelligent Systems*, J. C. Bansal, M. K. Gupta, H. Sharma, and B. Agarwal, Eds. Singapore: Springer, 2020, pp. 89–97.

[67] M. S. U. Islam, A. Kumar, and Y.-C. Hu, "Context-aware scheduling in fog computing: A survey, taxonomy, challenges and future directions," *J. Netw. Comput. Appl.*, vol. 180, Apr. 2021, Art. no. 103008.

[68] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3098–3130, 4th Quart. 2018.

[69] Z. Piao, M. Peng, Y. Liu, and M. Daneshmand, "Recent advances of edge cache in radio access networks for Internet of Things: Techniques, performances, and challenges," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 1010–1028, Feb. 2019.

[70] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1608–1631, Aug. 2019.

[71] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[72] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for Internet of Things realization," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2961–2991, 4th Quart., 2018.

[73] N. Hassan, K.-L. A. Yau, and C. Wu, "Edge computing in 5G: A review," *IEEE Access*, vol. 7, pp. 127276–127289, 2019.

[74] Y. Zhao, W. Wang, Y. Li, C. C. Meixner, M. Tornatore, and J. Zhang, "Edge computing and networking: A survey on infrastructures and applications," *IEEE Access*, vol. 7, pp. 101213–101230, 2019.

[75] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, and M. Guizani, "Multi-access edge computing: A survey," *IEEE Access*, vol. 8, pp. 197017–197046, 2020.

[76] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[77] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6722–6747, Aug. 2020.

[78] S. Keele, "Guidelines for performing systematic literature reviews in software engineering version 2.3," EBSE, Goyang-si, South Korea, EBSE Tech. Rep. EBSE-2007-01, 2007.

[79] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Inf. Softw. Technol.*, vol. 55, no. 12, pp. 2049–2075, 2013.

[80] P. Neves, R. Calé, M. Costa, G. Gaspar, J. Alcaraz-Calero, Q. Wang, J. Nightingale, G. Bernini, G. Carrozzo, Á. Valdivieso, L. J. García Villalba, M. Barros, A. Gravas, J. Santos, R. Maia, and R. Preto, "Future mode of operations for 5G—The SELFNET approach enabled by SDN/NFV," *Comput. Standards Interfaces*, vol. 54, pp. 229–246, Nov. 2017.

[81] C.-K. Huang, S.-H. Shen, C.-Y. Huang, T.-L. Chin, and C.-A. Shen, "S-cache: Toward an low latency service caching for edge clouds," in *Proc. ACM MobiHoc Workshop Pervasive Syst. IoT Era*, 2019, pp. 49–54.

[82] H. Ma, Z. Zhou, and X. Chen, "Leveraging the power of prediction: Predictive service placement for latency-sensitive mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6454–6468, Oct. 2020.

[83] S. Ma, S. Guo, K. Wang, W. Jia, and M. Guo, "A cyclic game for joint cooperation and competition of edge resource allocation," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 503–513.

[84] L. Chen and J. Xu, "Socially trusted collaborative edge computing in ultra dense networks," in *Proc. 2nd ACM/IEEE Symp. Edge Comput. (SEC)*. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 1–11.

[85] B. Yin, Y. Cheng, L. X. Cai, and X. Cao, "A cooperative edge computing scheme for reducing the cost of transferring big data in 5G networks," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 700–706.

[86] S. S. C. G., V. Chamola, C.-K. Tham, G. S., and N. Ansari, "An optimal delay aware task assignment scheme for wireless SDN networked edge cloudlets," *Future Gener. Comput. Syst.*, vol. 102, pp. 862–875, Jan. 2020.

[87] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[88] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[89] P. Zheng, A. Narayanan, and Z.-L. Zhang, "A closer look at NFV execution models," in *Proc. 3rd Asia–Pacific Workshop Netw. (APNet)*. New York, NY, USA: Association for Computing Machinery, Aug. 2019, pp. 85–91.

[90] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and availability driven VNF placement in a MEC-NFV environment," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–7.

[91] B. Yang, W. K. Chai, G. Pavlou, and K. V. Katsaros, "Seamless support of low latency mobile applications with NFV-enabled mobile edge-cloud," in *Proc. 5th IEEE Int. Conf. Cloud Netw. (Cloudnet)*, Oct. 2016, pp. 136–141.

[92] V. Sciancalepore, F. Giust, K. Samdanis, and Z. Yousaf, "A double-tier MEC-NFV architecture: Design and optimisation," in *Proc. IEEE Conf. Standards Commun. Netw. (CSCN)*, Oct. 2016, pp. 1–6.

[93] F. Chiti, R. Fantacci, F. Paganelli, and B. Picano, "Virtual functions placement with time constraints in fog computing: A matching theory perspective," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 980–989, Sep. 2019.

[94] D. Bhamare, A. Erbad, R. Jain, M. Zolanvari, and M. Samaka, "Efficient virtual network function placement strategies for cloud radio access networks," *Comput. Commun.*, vol. 127, pp. 50–60, Sep. 2018.

[95] M. Jia, W. Liang, and Z. Xu, "QoS-aware task offloading in distributed cloudlets with virtual network function services," in *Proc. 20th ACM Int. Conf. Modelling, Anal. Simulation Wireless Mobile Syst.*, Nov. 2017, pp. 109–116.

[96] T. Subramanya and R. Riggio, "Machine learning-driven scaling and placement of virtual network functions at the network edges," in *Proc. IEEE Conf. Netw. Softwarization (NetSoft)*, Jun. 2019, pp. 414–422.

[97] K. Yu, S. Eum, T. Kurita, Q. Hua, T. Sato, H. Nakazato, T. Asami, and V. P. Kafle, "Information-centric networking: Research and standardization status," *IEEE Access*, vol. 7, pp. 126164–126176, 2019.

[98] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Resource allocation for information-centric virtualized heterogeneous networks with in-network caching and mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 11339–11351, Dec. 2017.

[99] A. Afanasyev, J. Burke, T. Refaei, L. Wang, B. Zhang, and L. Zhang, "A brief introduction to named data networking," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2018, pp. 1–6.

[100] M. Amadeo, C. Campolo, and A. Molinaro, "NDNe: Enhancing named data networking to support cloudification at the edge," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2264–2267, Nov. 2016.

[101] A. E. C. Redondi, A. Arcia-Moret, and P. Manzoni, "Towards a scaled IoT pub/sub architecture for 5G networks: The case of multiaccess edge computing," in *Proc. IEEE 5th World Forum Internet Things (WF-IoT)*, Apr. 2019, pp. 436–441.

[102] G. Gür, P. Porambage, and M. Liyanage, "Convergence of ICN and MEC for 5G: Opportunities and challenges," *IEEE Commun. Standards Mag.*, vol. 4, no. 4, pp. 64–71, Dec. 2020.

[103] D. Grewe, M. Wagner, M. Arumaithurai, I. Psaras, and D. Kutscher, "Information-centric mobile edge computing for connected vehicle environments: Challenges and research directions," in *Proc. Workshop Mobile Edge Commun. (MECOMM)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 7–12.

[104] Y. He, C. Liang, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, and Y. Zhang, "Resource allocation in software-defined and information-centric vehicular networks with mobile edge computing," in *Proc. IEEE 86th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2017, pp. 1–5.

[105] H. Ben-Ammar and Y. Ghamri-Doudane, "An ICN-based approach for service caching in edge/fog environments," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[106] *Description of Network Slicing Concept*, NGMN Alliance, Frankfurt, Germany, 2016, pp. 1–7, vol. 1, no. 1.

[107] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[108] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.

[109] H. Zhang, N. Liu, X. Chu, K. Long, A. Aghvami, and V. C. M. Leung, "Network slicing based 5G and future mobile networks: Mobility, resource management, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 138–145, Aug. 2017.

[110] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Comput. Netw.*, vol. 167, Feb. 2020, Art. no. 106984.

[111] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72–79, May 2017.

[112] A. Ksentini and P. A. Frangoudis, "Toward slicing-enabled multi-access edge computing in 5G," *IEEE Netw.*, vol. 34, no. 2, pp. 99–105, Mar. 2020.

[113] Q. Xie, Q. Wang, N. Yu, H. Huang, and X. Jia, "Dynamic service caching in mobile edge networks," in *Proc. IEEE 15th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2018, pp. 73–79.

[114] A. Samanta and Z. Chang, "Adaptive service offloading for revenue maximization in mobile edge computing with delay-constraint," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3864–3872, Apr. 2019.

[115] L. Chen and J. Xu, "Budget-constrained edge service provisioning with demand estimation via bandit learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2364–2376, Oct. 2019.

[116] L. Chen, J. Xu, S. Ren, and P. Zhou, "Spatio–temporal edge service placement: A bandit learning approach," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 8388–8401, Dec. 2018.

[117] Y. Chen, S. Deng, H. Zhao, Q. He, Y. Li, and H. Gao, "Data-intensive application deployment at edge: A deep reinforcement learning approach," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2019, pp. 355–359.

[118] W. He, D. He, Y. Huang, Y. Zhang, Y. Xu, G. Yun-Feng, and W. Zhang, "Bandit learning-based service placement and resource allocation for mobile edge computing," in *Proc. 31st Annu. Int. Symp. Pers., Indoor Mobile Radio Commun.*, 2020, pp. 1–6.

[119] Z. Lei, H. Xu, L. Huang, and Z. Meng, "Joint service placement and request scheduling for multi-SP mobile edge computing network," in *Proc. IEEE 26th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2020, pp. 27–34.

[120] A. Morgado, K. M. S. Huq, S. Mumtaz, and J. Rodriguez, "A survey of 5G technologies: Regulatory, standardization and industrial perspectives," *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 87–97, Apr. 2018.

[121] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Mar. 2012, pp. 1107–1115.

[122] N. W. Sung, N. T. Pham, T. Huynh, W. J. Hwang, I. You, and K. R. Choo, "Prediction-based association control scheme in dense femtocell networks," *PLoS ONE*, vol. 12, no. 3, pp. 1–23, Mar. 2017.

[123] Y. Liang, J. Ge, S. Zhang, J. Wu, Z. Tang, and B. Luo, "A utility-based optimization framework for edge service entity caching," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 11, pp. 2384–2395, Nov. 2019.

[124] 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.

[125] M. F. Hossain, A. U. Mahin, T. Debnath, F. B. Mosharrof, and K. Z. Islam, "Recent research in cloud radio access network (C-RAN) for 5G cellular systems—A survey," *J. Netw. Comput. Appl.*, vol. 139, pp. 31–48, Aug. 2019.

[126] M. A. Habibi, M. Nasimi, B. Han, and H. D. Schotten, "A comprehensive survey of RAN architectures toward 5G mobile communication system," *IEEE Access*, vol. 7, pp. 70371–70421, 2019.

[127] J. Tang, R. Wen, T. Q. S. Quek, and M. Peng, "Fully exploiting cloud computing to achieve a green and flexible C-RAN," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 40–46, Nov. 2017.

[128] J. Tang and T. Q. S. Quek, "The role of cloud computing in content-centric mobile networking," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 52–59, Aug. 2016.

[129] X. Li, X. Wang, K. Li, and V. C. M. Leung, "CaaS: Caching as a service for 5G networks," *IEEE Access*, vol. 5, pp. 5982–5993, 2017.

[130] M. Peng, Y. Li, Z. Zhao, and C. Wang, "System architecture and key technologies for 5G heterogeneous cloud radio access networks," *IEEE Netw.*, vol. 29, no. 2, pp. 6–14, Mar./Apr. 2015.

[131] S. Dash, B. J. R. Sahu, and N. Saxena, "Social network aware caching for 5G radio access network," *IETE Tech. Rev.*, vol. 34, no. 1, pp. 52–61, Dec. 2017.

[132] W. Huang, Y. Huang, S. He, and L. Yang, "Cloud and edge multicast beamforming for cache-enabled ultra-dense networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3481–3485, Mar. 2020.

[133] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 3, no. 1, pp. 70–95, Feb. 2016.

[134] M. Van Steen and A. S. Tanenbaum, *Distributed Systems*. Leiden, The Netherlands: Maarten van Steen, 2017.

[135] H. T. Malazi and S. Clarke, "Distributed service placement and workload orchestration in a multi-access edge computing environment," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Sep. 2021, pp. 241–251.

[136] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 1281–1290.

[137] O. Ascigil, T. K. Phan, A. G. Tasiopoulos, V. Sourlas, I. Psaras, and G. Pavlou, "On uncoordinated service placement in edge-clouds," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2017, pp. 41–48.

[138] I. Farris, T. Taleb, A. Iera, and H. Flinck, "Lightweight service replication for ultra-short latency applications in mobile edge networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[139] L. Gillam, K. Katsaros, M. Dianati, and A. Mouzakitis, "Exploring edges for connected and autonomous driving," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 148–153.

[140] K. K. Breitman, M. A. Casanova, and W. Truszkowski, "Software agents," in *Semantic Web: Concepts, Technologies and Applications*. London, U.K.: Springer, 2007, pp. 219–228, doi: 10.1007/978-1-84628-710-7_11.

[141] P. Bagga and R. Hans, "Mobile agents system security: A systematic survey," *ACM Comput. Surv.*, vol. 50, no. 5, pp. 1–45, Sep. 2018.

[142] O. Urra, S. Ilarri, R. Trillo, and E. Mena, "Mobile software agents for mobile applications," in *Handbook of Research on Mobile Software Engineering: Design, Implementation, and Emergent Applications*. Hershey, PA, USA: IGI Global, 2012, pp. 725–740.

[143] A. Carzaniga, G. P. Picco, and G. Vigna, "Is code still moving around? Looking back at a decade of code mobility," in *Proc. 29th Int. Conf. Softw. Eng. (ICSE Companion)*, May 2007, pp. 9–20.

[144] P. Angin, B. Bhargava, and Z. Jin, "A self-cloning agents based model for high-performance mobile-cloud computing," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun. 2015, pp. 301–308.

[145] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Gener. Comput. Syst.*, vol. 79, pp. 849–861, Feb. 2018.

[146] A. Palade, A. Mukhopadhyay, A. Kazmi, C. Cabrera, E. Nomayo, G. Iosifidis, M. Ruffini, and S. Clarke, "A swarm-based approach for function placement in federated edges," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Nov. 2020, pp. 48–50.

[147] S. R. Chaudhry, A. Palade, A. Kazmi, and S. Clarke, "Improved QoS at the edge using serverless computing to deploy virtual network functions," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10673–10683, Oct. 2020.

[148] C. Cicconetti, M. Conti, and A. Passarella, "Architecture and performance evaluation of distributed computation offloading in edge computing," *Simul. Model. Pract. Theory*, vol. 101, May 2020, Art. no. 102007.

[149] A. Palade, A. Kazmi, and S. Clarke, "An evaluation of open source serverless computing frameworks support at the edge," in *Proc. IEEE World Congr. Services (SERVICES)*, Jul. 2019, pp. 206–211.

[150] S. Luo, Y. Wen, W. Xu, and D. Puthal, "Adaptive task offloading auction for industrial CPS in mobile edge computing," *IEEE Access*, vol. 7, pp. 169055–169065, 2019.

[151] Q. Zhang, Q. Zhu, M. F. Zhani, R. Boutaba, and J. L. Hellerste, "Dynamic service placement in geographically distributed clouds," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 762–772, Dec. 2013.

[152] B. Németh, M. Szalay, J. Dóka, M. Rost, S. Schmid, L. Toka, and B. Sonkoly, "Fast and efficient network service embedding method with adaptive offloading to the edge," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 178–183.

[153] K. Peng, J. Nie, N. Kumar, C. Cai, J. Kang, Z. Xiong, and Y. Zhang, "Joint optimization of service chain caching and task offloading in mobile edge computing," *Appl. Soft Comput.*, vol. 103, May 2021, Art. no. 107142.

[154] R. S. Prakash, N. Karamchandani, V. Kavitha, and S. Moharir, "Partial service caching at the edge," in *Proc. 18th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOPT)*, 2020, pp. 1–8.

[155] V. Farhadi, F. Mehmeti, T. He, T. L. Porta, H. Khamfroush, S. Wang, and K. S. Chan, "Service placement and request scheduling for data-intensive applications in edge clouds," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1279–1287.

[156] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, "A multi-objective service placement and load distribution in edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–7.

[157] Y. Li, W. Dai, X. Gan, H. Jin, L. Fu, H. Ma, and X. Wang, "Cooperative service placement and scheduling in edge clouds: A deadline-driven approach," *IEEE Trans. Mobile Comput.*, early access, Feb. 23, 2021, doi: 10.1109/TMC.2021.3061602.

[158] J. Kwak and G. Iosifidis, "DSP: Dynamic service placement with reconfiguration cost and delay," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2020, pp. 244–247.

[159] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, "Dynamic service placement and load distribution in edge computing," in *Proc. 16th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2020, pp. 1–9.

[160] A. G. Tasiopoulos, O. Ascigil, I. Psaras, and G. Pavlou, "Edge-map: Auction markets for edge resource provisioning," in *Proc. IEEE 19th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2018, pp. 14–22.

[161] W. Ni, H. Tian, X. Lyu, and S. Fan, "Service-dependent task offloading for multiuser mobile edge computing system," *Electron. Lett.*, vol. 55, no. 15, pp. 839–841, Jul. 2019.

[162] Z. Xu, L. Zhou, S. Chi-Kin Chau, W. Liang, Q. Xia, and P. Zhou, "Collaborate or separate? Distributed service caching in mobile edge clouds," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 2066–2075.

[163] J. Yan, S. Bi, L. Duan, and Y.-J.-A. Zhang, "Pricing-driven service caching and task offloading in mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4495–4512, Jul. 2021.

[164] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, and H. Zhang, "Incentive mechanism for computation offloading using edge computing: A Stackelberg game approach," *Comput. Netw.*, vol. 129, pp. 399–409, Dec. 2017.

[165] V. C. L. Narayana, S. Moharir, and N. Karamchandani, ''RetroRenting: An online policy for service caching at the edge,'' in *Proc. 18th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOPT)*, 2020, pp. 1–8.

[166] B. Gao, Z. Zhou, F. Liu, and F. Xu, ''Winning at the starting line: Joint network selection and service placement for mobile edge computing,'' in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1459–1467.

[167] C. Sonmez, A. Ozgovde, and C. Ersoy, ''Fuzzy workload orchestration for edge computing,'' *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 769–782, Jun. 2019.

[168] N. Yu, Q. Xie, Q. Wang, H. Du, H. Huang, and X. Jia, ''Collaborative service placement for mobile edge computing applications,'' in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[169] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, ''Dynamic service placement for mobile micro-clouds with predicted future costs,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, Apr. 2017.

[170] Y. Zhang, L. Jiao, J. Yan, and X. Lin, ''Dynamic service placement for virtual reality group gaming on mobile edge cloudlets,'' *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1881–1897, Aug. 2019.

[171] J. Xu, L. Chen, and P. Zhou, ''Joint service caching and task offloading for mobile edge computing in dense networks,'' in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 207–215.

[172] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, ''Joint service placement and request routing in multi-cell mobile edge computing networks,'' in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 10–18.

[173] M. S. Elbamby, M. Bennis, and W. Saad, ''Proactive edge computing in latency-constrained fog networks,'' in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2017, pp. 1–6.

[174] L. Baresi, D. F. Mendonça, and G. Quattrocchi, ''PAPS: A framework for decentralized self-management at the edge,'' in *Proc. Int. Conf. Service-Oriented Comput.* Cham, Switzerland: Springer, 2019, pp. 508–522.

[175] D. Lan, A. Taherkordi, F. Eliassen, and L. Liu, ''Deep reinforcement learning for computation offloading and caching in fog-based vehicular networks,'' in *Proc. IEEE 17th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Dec. 2020, pp. 622–630.

[176] L. Li and H. Zhang, ''Delay optimization strategy for service cache and task offloading in three-tier architecture mobile edge computing system,'' *IEEE Access*, vol. 8, pp. 170211–170224, 2020.

[177] R. Wang, Y. Cao, A. Noor, T. A. Alamoudi, and R. Nour, ''Agent-enabled task offloading in UAV-aided mobile edge computing,'' *Comput. Commun.*, vol. 149, pp. 324–331, Jan. 2020.

[178] C. Guerrero, I. Lera, and C. Juiz, ''A lightweight decentralized service placement policy for performance optimization in fog computing,'' *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 6, pp. 2435–2452, 2019.

[179] H. Zhao, S. Deng, Z. Liu, J. Yin, and S. Dustdar, ''Distributed redundancy scheduling for microservice-based applications at the edge,'' *IEEE Trans. Services Comput.*, early access, Aug. 3, 2020, doi: 10.1109/TSC.2020.3013600.

[180] H. Xing, J. Cui, Y. Deng, and A. Nallanathan, ''Energy-efficient proactive caching for fog computing with correlated task arrivals,'' in *Proc. IEEE 20th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jul. 2019, pp. 1–5.

[181] X. He, Z. Tu, X. Xu, and Z. Wang, ''Re-deploying microservices in edge and cloud environment for the optimization of user-perceived service quality,'' in *Service-Oriented Computing*, S. Yangui, I. B. Rodriguez, K. Drira, and Z. Tari, Eds. Cham, Switzerland: Springer, 2019, pp. 555–560.

[182] A. Varasteh, S. Hofmann, N. Deric, M. He, D. Schupke, W. Kellerer, and C. M. Machuca, ''Mobility-aware joint service placement and routing in space-air-ground integrated networks,'' in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[183] M. Li, L. Rui, X. Qiu, S. Guo, and X. Yu, ''Design of a service caching and task offloading mechanism in smart grid edge network,'' in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 249–254.

[184] T. He, H. Khamfroush, S. Wang, T. La Porta, and S. Stein, ''It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources,'' in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 365–375.

[185] J. Li, H. Zhang, H. Ji, and X. Li, ''Joint computation offloading and service caching for MEC in multi-access networks,'' in *Proc. IEEE 30th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2019, pp. 1–6.

[186] J. Xu, K. Ota, and M. Dong, ''Plug-and-play for fog: Dynamic service placement in wireless multimedia networks,'' in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Aug. 2018, pp. 490–494.

[187] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, ''Towards QoS-aware fog service placement,'' in *Proc. IEEE 1st Int. Conf. Fog Edge Comput. (ICFEC)*, May 2017, pp. 89–96.

[188] Y.-J. Yu, T.-C. Chiu, A.-C. Pang, M.-F. Chen, and J. Liu, ''Virtual machine placement for backhaul traffic minimization in fog radio access networks,'' in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–7.

[189] L. Gu, ''Cost efficient resource management in fog computing supported medical cyber-physical system,'' *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 108–119, Dec. 2017.

[190] Q. Liu, S. Huang, J. Opadere, and T. Han, ''An edge network orchestrator for mobile augmented reality,'' in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 756–764.

[191] S. B. Nath, S. Chattopadhyay, R. Karmakar, S. K. Addya, S. Chakraborty, and S. K. Ghosh, ''PTC: Pick-test-choose to place containerized microservices in IoT,'' in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[192] X. Ma, A. Zhou, S. Zhang, and S. Wang, ''Cooperative service caching and workload scheduling in mobile edge computing,'' in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 2076–2085.

[193] Z. Xiang, S. Deng, J. Taheri, and A. Zomaya, ''Dynamical service deployment and replacement in resource-constrained edges,'' *Mobile Netw. Appl.*, vol. 25, no. 2, pp. 674–689, Apr. 2020.

[194] T. Zhao, I.-H. Hou, S. Wang, and K. Chan, ''Red/LeD: An asymptotically optimal and scalable online algorithm for service caching at the edge,'' *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1857–1870, Aug. 2018.

[195] Q. Xia, W. Liang, Z. Xu, and B. Zhou, ''Online algorithms for location-aware task offloading in two-tiered mobile cloud environments,'' in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput.*, Dec. 2014, pp. 109–116.

[196] X. Jie, T. Liu, H. Gao, C. Cao, P. Wang, and W. Tong, ''A DQN-based approach for online service placement in mobile edge computing,'' in *Proc. Int. Conf. Collaborative Comput., Netw., Appl. Worksharing*. Cham, Switzerland: Springer, 2020, pp. 169–183.

[197] H. Hao, C. Xu, L. Zhong, and G.-M. Muntean, ''A multi-update deep reinforcement learning algorithm for edge computing service offloading,'' in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 3256–3264.

[198] S. He, X. Lyu, W. Ni, H. Tian, R. P. Liu, and E. Hossain, ''Virtual service placement for edge computing under finite memory and bandwidth,'' *IEEE Trans. Commun.*, vol. 68, no. 12, pp. 7702–7718, Dec. 2020.

[199] L. Wang, L. Jiao, T. He, J. Li, and M. Muhlhauser, ''Service entity placement for social virtual reality applications in edge computing,'' in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 468–476.

[200] L. Wang, L. Jiao, T. He, J. Li, and H. Bal, ''Service placement for collaborative edge applications,'' *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 34–47, Feb. 2021.

[201] V. Chamola, C.-K. Tham, and G. S. S. Chalapathi, ''Latency aware mobile task assignment and load balancing for edge cloudlets,'' in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2017, pp. 587–592.

[202] A. J. Fahs and G. Pierre, ''Tail-latency-aware fog application replica placement,'' in *Proc. Int. Conf. Service-Oriented Comput.* Cham, Switzerland: Springer, 2020, pp. 508–524.

[203] P. Zhao, P. Wang, X. Yang, and J. Lin, ''Towards cost-efficient edge intelligent computing with elastic deployment of container-based microservices,'' *IEEE Access*, vol. 8, pp. 102947–102957, 2020.

[204] H. Ma, Z. Zhou, and X. Chen, ''Predictive service placement in mobile edge computing,'' in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Aug. 2019, pp. 792–797.

[205] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, ''Adaptive user-managed service placement for mobile edge computing: An online learning approach,'' in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1468–1476.

[206] H. Wei, H. Luo, and Y. Sun, ''Mobility-aware service caching in mobile edge computing for Internet of Things,'' *Sensors*, vol. 20, no. 3, p. 610, Jan. 2020.

[207] A. Aral, I. Brandic, R. B. Uriarte, R. De Nicola, and V. Scoca, ''Addressing application latency requirements through edge scheduling,'' *J. Grid Comput.*, vol. 17, no. 4, pp. 677–698, Dec. 2019.

[208] Y. Qian, L. Hu, J. Chen, X. Guan, M. M. Hassan, and A. Alelaiwi, ''Privacy-aware service placement for mobile edge computing via federated learning,'' *Inf. Sci.*, vol. 505, pp. 562–570, Dec. 2019.
[209] T. Bahreini and D. Grosu, ''Efficient placement of multi-component applications in edge computing systems,'' in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 1–11.
[210] Z. Ning, P. Dong, X. Wang, S. Wang, X. Hu, S. Guo, T. Qiu, B. Hu, and R. Y. K. Kwok, ''Distributed and dynamic service placement in pervasive edge computing networks,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 6, pp. 1277–1292, Jun. 2021.
[211] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, ''Towards a better understanding of context and context-awareness,'' in *Proc. Int. Symp. Handheld Ubiquitous Comput.* Berlin, Germany: Springer, 1999, pp. 304–307.
[212] M. Baldauf, S. Dustdar, and F. Rosenberg, ''A survey on context-aware systems,'' *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 2, no. 4, pp. 263–277, 2007.
[213] S. B. Hassanpour, A. Diyanat, A. Khonsari, S. P. Shariatpanahi, and A. Dadlani, ''Context-aware privacy preservation in network caching: An information theoretic approach,'' *IEEE Commun. Lett.*, vol. 25, no. 1, pp. 54–58, Jan. 2021.
[214] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, ''Context aware computing for the Internet of Things: A survey,'' *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 414–454, 1st Quart., 2014.
[215] S. Deng, Z. Xiang, J. Taheri, M. A. Khoshkholghi, J. Yin, A. Y. Zomaya, and S. Dustdar, ''Optimal application deployment in resource constrained distributed edges,'' *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 1907–1923, May 2021.
[216] W. Zhang, J. Chen, Y. Zhang, and D. Raychaudhuri, ''Towards efficient edge cloud augmentation for virtual reality MMOGs,'' in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 1–14.
[217] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, ''Joint load balancing and offloading in vehicular edge computing and networks,'' *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.
[218] S. Liu, L. Liu, J. Tang, B. Yu, and W. Shi, ''Edge computing for autonomous driving: Opportunities and challenges,'' *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Jun. 2019.
[219] S. Raza, S. Wang, M. Ahmed, and M. R. Anwar, ''A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions,'' *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–19, Feb. 2019.
[220] D. Lin, S. Hu, Y. Gao, and Y. Tang, ''Optimizing MEC networks for healthcare applications in 5G communications with the authenticity of users' priorities,'' *IEEE Access*, vol. 7, pp. 88592–88600, 2019.
[221] A. B. De Souza, P. A. L. Rego, T. Carneiro, J. D. C. Rodrigues, P. P. R. Filho, J. N. De Souza, V. Chamola, V. H. C. De Albuquerque, and B. Sikdar, ''Computation offloading for vehicular environments: A survey,'' *IEEE Access*, vol. 8, pp. 198214–198243, 2020.
[222] A. Palade and S. Clarke, ''Collaborative agent communities for resilient service composition in mobile environments,'' *IEEE Trans. Services Comput.*, early access, Jan. 8, 2020, doi: 10.1109/TSC.2020.2964753.
[223] C. Cabrera, A. Palade, G. White, and S. Clarke, ''Services in IoT: A service planning model based on consumer feedback,'' in *Proc. Int. Conf. Service-Oriented Comput.* Cham, Switzerland: Springer, 2018, pp. 304–313.
[224] B. Yi, X. Wang, S. K. Das, K. Li, and M. Huang, ''A comprehensive survey of network function virtualization,'' *Comput. Netw.*, vol. 133, pp. 212–262, Mar. 2018.
[225] I. Alam, K. Sharif, F. Li, Z. Latif, M. M. Karim, S. Biswas, B. Nour, and Y. Wang, ''A survey of network virtualization techniques for Internet of Things using SDN and NFV,'' *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–40, Mar. 2021.
[226] G. Sun, Y. Li, Y. Li, D. Liao, and V. Chang, ''Low-latency orchestration for workflow-oriented service function chain in edge computing,'' *Future Gener. Comput. Syst.*, vol. 85, pp. 116–128, Aug. 2018.
[227] L. Askari, A. Hmaity, F. Musumeci, and M. Tornatore, ''Virtual-network-function placement for dynamic service chaining in metro-area networks,'' in *Proc. Int. Conf. Opt. Netw. Design Modeling (ONDM)*, May 2018, pp. 136–141.
[228] B. Donassolo, I. Fajjari, A. Legrand, and P. Mertikopoulos, ''Load aware provisioning of IoT services on fog computing platform,'' in *Proc. IEEE Int. Conf. Commun. (ICC)*, Mar. 2019, pp. 1–7.
[229] T. Djemai, P. Stolf, T. Monteil, and J.-M. Pierson, ''A discrete particle swarm optimization approach for energy-efficient IoT services placement over fog infrastructures,'' in *Proc. 18th Int. Symp. Parallel Distrib. Comput. (ISPDC)*, Jun. 2019, pp. 32–40.
[230] L. Liu, H. Tan, S. H.-C. Jiang, Z. Han, X.-Y. Li, and H. Huang, ''Dependent task placement and scheduling with function configuration in edge computing,'' in *Proc. Int. Symp. Quality Service*, Jun. 2019, pp. 1–10.
[231] M. Taneja and A. Davy, ''Resource aware placement of IoT application modules in fog-cloud computing paradigm,'' in *Proc. IEEE Symp. Integr. Netw. Service Manage.*, May 2017, pp. 1222–1228.
[232] G. Solmaz and D. Turgut, ''A survey of human mobility models,'' *IEEE Access*, vol. 7, pp. 125711–125731, 2019.
[233] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, ''Optimized placement of scalable IoT services in edge computing,'' in *Proc. IEEE Symp. Integr. Netw. Service Manage.*, Apr. 2019, pp. 189–197.
[234] H. D. Chantre and N. L. S. da Fonseca, ''Redundant placement of virtualized network functions for LTE evolved multimedia broadcast multicast services,'' in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–7.
[235] B. Li, Q. He, G. Cui, X. Xia, F. Chen, H. Jin, and Y. Yang, ''READ: Robustness-oriented edge application deployment in edge computing environment,'' *IEEE Trans. Services Comput.*, early access, Aug. 10, 2021, doi: 10.1109/TSC.2020.3015316.
[236] H. Huang, H. Zhang, T. Guo, J. Guo, and C. He, ''Reliable redundant services placement in federated micro-clouds,'' in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2019, pp. 446–453.
[237] Z. Nezami, K. Zamanifar, K. Djemame, and E. Pournaras, ''Decentralized edge-to-cloud load balancing: Service placement for the Internet of Things,'' *IEEE Access*, vol. 9, pp. 64983–65000, 2021.
[238] J. Famaey, T. Wauters, F. De Turck, B. Dhoedt, and P. Demeester, ''Network-aware service placement and selection algorithms on large-scale overlay networks,'' *Comput. Commun.*, vol. 34, no. 15, pp. 1777–1787, Sep. 2011.
[239] G. White and S. Clarke, ''Short-term QoS forecasting at the edge for reliable service applications,'' *IEEE Trans. Services Comput.*, early access, Feb. 24, 2020, doi: 10.1109/TSC.2020.2975799.
[240] K. Velasquez, D. P. Abreu, M. R. M. Assis, C. Senna, D. F. Aranha, L. F. Bittencourt, N. Laranjeiro, M. Curado, M. Vieira, E. Monteiro, and E. Madeira, ''Fog orchestration for the Internet of Everything: State-of-the-art and research challenges,'' *J. Internet Services Appl.*, vol. 9, no. 1, p. 14, Dec. 2018.
[241] C. Cabrera and S. Clarke, ''A self-adaptive service discovery model for smart cities,'' *IEEE Trans. Services Comput.*, vol. 15, no. 1, pp. 386–399, Jan. 2022.
[242] C. Cabrera, A. Palade, G. White, and S. Clarke, ''An urban-driven service request management model,'' in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2020, pp. 1–7.
[243] B. Khazael, H. T. Malazi, and S. Clarke, ''Complex event processing in smart city monitoring applications,'' *IEEE Access*, vol. 9, pp. 143150–143165, 2021.
[244] J. M. Bailey, H. T. Malazi, and S. Clarke, ''Smoothing speed variability in age-friendly urban traffic management,'' in *Proc. Int. Conf. Comput. Sci. (ICCS)*, 2021, pp. 3–16.
[245] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, ''Edge-enabled V2X service placement for intelligent transportation systems,'' *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1380–1392, Apr. 2021.
[246] S. Wang, K. Chan, R. Urgaonkar, T. He, and K. K. Leung, ''Emulation-based study of dynamic service placement in mobile micro-clouds,'' in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2015, pp. 1046–1051.
[247] S. Forti, A. Pagiaro, and A. Brogi, ''Simulating FogDirector application management,'' *Simul. Model. Pract. Theory*, vol. 101, May 2020, Art. no. 102021.
[248] O. Skarlat, V. Karagiannis, T. Rausch, K. Bachmann, and S. Schulte, ''A framework for optimization, service placement, and runtime operation in the fog,'' in *Proc. IEEE/ACM 11th Int. Conf. Utility Cloud Comput. (UCC)*, Dec. 2018, pp. 164–173.
[249] H. A. Alharbi, T. E. H. El-Gorashi, and J. M. H. Elmirghani, ''Energy efficient virtual machine services placement in cloud-fog architecture,'' in *Proc. 21st Int. Conf. Transparent Opt. Netw. (ICTON)*, Jul. 2019, pp. 1–6.
[250] D. N. Jha, K. Alwasel, A. Alshoshan, X. Huang, R. K. Naha, S. K. Battula, S. Garg, D. Puthal, P. James, A. Y. Zomaya, S. Dustdar, and R. Ranjan, ''IoTSim-edge: A simulation framework for modeling the behaviour of IoT and edge computing environments,'' 2019, *arXiv:1910.03026*.
[251] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. Rose, and R. Buyya, ''CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,'' *Softw., Pract. Exper*, vol. 41, no. 1, pp. 23–50, Aug. 2010.

[252] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, ''ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers,'' *Softw., Pract. Exper.*, vol. 47, no. 4, pp. 505–521, Apr. 2017.

[253] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, ''iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments,'' *Softw., Pract. Exper.*, vol. 47, no. 9, pp. 1275–1296, 2017.

[254] M. R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, ''IFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments,'' 2021, *arXiv:2109.05636*.

[255] C. Sonmez, A. Ozgovde, and C. Ersoy, ''EdgeCloudSim: An environment for performance evaluation of edge computing systems,'' *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, p. e3493, Nov. 2018.

[256] A. Varga and R. Hornig, ''An overview of the OMNeT++ simulation environment,'' in *Proc. 1st Int. ICST Conf. Simulation Tools Techn. Commun. Netw. Syst.*, 2008, pp. 1–10.

[257] *OMNeT++: Discrete Event Simulator*. Accessed: Aug. 2020. [Online]. Available: https://omnetpp.org

[258] T. Qayyum, A. W. Malik, M. A. K. Khattak, O. Khalid, and S. U. Khan, ''FogNetSim++: A toolkit for modeling and simulation of distributed fog environment,'' *IEEE Access*, vol. 6, pp. 63570–63583, 2018.

[259] A. Brogi, S. Forti, and A. Ibrahim, ''How to best deploy your fog applications, probably,'' in *Proc. IEEE 1st Int. Conf. Fog Edge Comput. (ICFEC)*, May 2017, pp. 105–114.

[260] I. Lera, C. Guerrero, and C. Juiz, ''YAFS: A simulator for IoT scenarios in fog computing,'' *IEEE Access*, vol. 7, pp. 91745–91758, 2019.

[261] I. Petri, O. Rana, A. R. Zamani, and Y. Rezgui, ''Edge-cloud orchestration: Strategies for service placement and enactment,'' in *Proc. IEEE Int. Conf. Cloud Eng. (ICE)*, Jun. 2019, pp. 67–75.

[262] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, ''PlanetLab: An overlay testbed for broad-coverage services,'' *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, Jul. 2003.

[263] M. Selimi, L. Cerdà-Alabern, F. Freitag, L. Veiga, A. Sathiaseelan, and J. Crowcroft, ''A lightweight service placement approach for community network micro-clouds,'' *J. Grid Comput.*, vol. 17, no. 1, pp. 169–189, 2018.

[264] M. R. Saniat, H. N. M. Quoc, H. Le Van, D. Le Phuoc, M. Serrano, and M. Hauswirth, ''Autonomic frameworks deployment using configuration and service delivery models for the Internet of Things,'' in *Interoperability and Open-Source Solutions for the Internet of Things*. Cham, Switzerland: Springer, 2015, pp. 89–102.

[265] A.-Y. Son and E.-N. Huh, ''Multi-objective service placement scheme based on fuzzy-AHP system for distributed cloud computing,'' *Appl. Sci.*, vol. 9, no. 17, p. 3550, Aug. 2019.

[266] *PlanetLab: An Open Platform for Developing, Deploying, and Accessing Planetary-Scale Services*. Accessed: Aug. 2020. [Online]. Available: https://www.planet-lab.org

[267] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, ''FIT IoT-LAB: A large scale open experimental IoT testbed,'' in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 459–464.

[268] *Grid'5000 is a Large-Scale and Flexible Testbed for Experimentdriven Research*. Accessed: Aug. 2020. [Online]. Available: https://www.grid5000.fr/w/Grid5000:Home

[269] *Silecs: Super Infrastructure for Large-Scale Experimental Computer Science*. Accessed: Aug. 2020. [Online]. Available: https://www.silecs.net

[270] *Fed4FIRE an EU Project Offering the Largest Federation Worldwide of Next Generation Internet (NGI) Testbeds*. Accessed: Aug. 2020. [Online]. Available: https://www.fed4fire.eu

[271] J. Choi and S. Ahn, ''Scalable service placement in the fog computing environment for the IoT-based smart city,'' *J. Inf. Process. Syst.*, vol. 15, no. 2, pp. 440–448, 2019.

[272] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, ''CRAWDAD data set epfl/mobility (v. 2009-02-24),'' CRAWDAD, Hanover, NH, USA, Tech. Rep. 2009-02-24, 2009.

[273] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and A. Y. Yang, ''Optimal edge user allocation in edge computing with variable sized vector bin packing,'' in *Service-Oriented Computing*, C. Pahl, M. Vukovic, J. Yin, and Q. Yu, Eds. Cham, Switzerland: Springer, 2018, pp. 230–245.

[274] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, ''A parsimonious model of mobile partitioned networks with clustering,'' in *Proc. 1st Int. Commun. Syst. Netw. Workshops*, 2009, pp. 1–10.

[275] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma, ''Understanding transportation modes based on GPS data for web applications,'' *ACM Trans. Web*, vol. 4, no. 1, pp. 1–36, Jan. 2010.

[276] M. Gramaglia, M. Fiore, A. Tarable, and A. Banchs, ''Preserving mobile subscriber privacy in open datasets of spatiotemporal trajectories,'' in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[277] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. J. Epema, ''The grid workloads archive,'' *Future Gener. Comput. Syst.*, vol. 24, no. 7, pp. 672–686, Jul. 2008.

[278] X. Cheng, C. Dale, and J. Liu. (2008). Dataset for 'statistics and social network of YouTube videos. Simon Fraster University, Burnaby, BC, Canada. [Online]. Available: http://netsg.cs.sfu.ca/youtubedata

[279] C. Reiss, J. Wilkes, and J. L. Hellerstein, ''Google cluster-usage traces: Format+ schema,'' Google, Mountain View, CA, USA, Tech. Rep. 2011.10.27, 2011, pp. 1–14.

[280] S. L. Lim, P. J. Bentley, N. Kanakam, F. Ishikawa, and S. Honiden, ''Investigating country differences in mobile app user behavior and challenges for software engineering,'' *IEEE Trans. Softw. Eng.*, vol. 41, no. 1, pp. 40–64, Jan. 2015.

[281] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, ''A multi-source dataset of urban life in the city of Milan and the province of trentino,'' *Sci. Data*, vol. 2, no. 1, pp. 1–15, Dec. 2015.

[282] *NYC Wi-Fi Hotspot Locations*. Accessed: Apr. 2021. [Online]. Available: https://data.world/city-of-ny/a9wemtpn

[283] S. Svorobej, P. T. Endo, M. Bendechache, C. Filelis-Papadopoulos, K. Giannoutakis, G. Gravvanis, D. Tzovaras, J. Byrne, and T. Lynn, ''Simulating fog and edge computing scenarios: An overview and research challenges,'' *Future Internet*, vol. 11, no. 3, p. 55, Feb. 2019.

[284] C. H. C. Jojoa, S. Svorobej, A. Palade, A. Kazmi, and S. Clarke, ''MAACO: A dynamic service placement model for smart cities,'' *IEEE Trans. Services Comput.*, early access, Jan. 13, 2022, doi: 10.1109/TSC.2022.3143029.

HADI TABATABAEE MALAZI received the Ph.D. degree in computer engineering (distributed systems) from the University of Isfahan. He is currently an Assistant Professor at Maynooth University. Before taking up this position, he worked as a Research Fellow at Trinity College Dublin in the enable smart city research program. He was also an Assistant Professor with the Computer Science and Engineering Faculty, Shahid Beheshti University, from 2013 to 2019. His main research interests include pervasive and edge computing and emphasizing smart service orchestration.

SAQIB RASOOL CHAUDHRY received the Ph.D. degree in communication and system engineering from Brunel University London, Uxbridge, U.K., in 2010. He worked as a Research Fellow with Dublin City University, Dublin, Ireland, in 2017, and Trinity College Dublin, Dublin, in 2019. He currently holds a position of Lecturer with the Department of Computer Science, C.I.T., Cork, Ireland. His primary research interests include FPGAs, M.E.C., ICN, and cyber security.

AQEEL KAZMI received the M.Sc. and Ph.D. degrees in computer science from University College Dublin, Dublin, Ireland, in 2009 and 2014, respectively. He is currently a Research Fellow with Trinity College Dublin, Dublin, Ireland. His research interests include ubiquitous computing, big data management and analytics, information visualization, the Internet of Things, and smart city applications.

ANDREI PALADE received the M.Sc. degree in computer science from the University of Glasgow, Glasgow, U.K., in 2014, and the Ph.D. degree in computer science from Trinity College Dublin, Dublin, Ireland, in 2019. He is currently a Postdoctoral Researcher with Trinity College Dublin. His research interests include multi-objective optimization, nature-inspired metaheuristics, and QoS-aware service composition for dynamic IoT environments.

CHRISTIAN CABRERA received the B.Sc. degree in systems engineering from the University of Nariño, Pasto, Colombia, the M.Sc. degree in computer science from Los Andes University, Bogot'a, Colombia, and the Ph.D. degree in computer science from Trinity College Dublin. He is currently a Research Associate at the University of Cambridge. His research interests include intersection of service oriented computing, data oriented architectures, and artificial intelligence.

GARY WHITE received the B.A.I., M.A.I., and Ph.D. degrees in computer engineering from Trinity College Dublin, Dublin, Ireland. He is currently a Postdoctoral Researcher with Trinity College Dublin. His research interests include QoS, machine learning, and edge computing. His research is funded by the Science Foundation Ireland (S.F.I.).

SIOBHÁN CLARKE received the B.Sc. and Ph.D. degrees in computer science from Dublin City University. She is currently a Professor and a fellow with Trinity College Dublin, where she leads the Distributed Systems Group and the Director of Future Cities, the Trinity Centre for Smart and Sustainable Cities. Her research interests include software engineering models for the dynamic provision of smart software services to urban stakeholders in large-scale and mobile environments.