Survey on Placement Methods in the Edge and Beyond

Balázs Sonkoly, János Czentye, Márk Szalay, Balázs Németh, and László Toka, *Member, IEEE*

***Abstract*—Edge computing is a (r)evolutionary extension of traditional cloud computing. It expands central cloud infrastructure with execution environments close to the users in terms of latency in order to enable a new generation of cloud applications. This paradigm shift has opened the door for telecommunications operators, mobile and fixed network vendors: they have joined the cloud ecosystem as essential stakeholders considerably influencing the future success of the technology. A key problem in edge computing is the optimal placement of computational units (virtual machines, containers, tasks or functions) of novel distributed applications. These components are deployed to a geographically distributed virtualized infrastructure and heterogeneous networking technologies are invoked to connect them while respecting quality requirements. The optimal hosting environment should be selected based on multiple criteria by novel scheduler algorithms which can cope with the new challenges of distributed cloud architecture where networking aspects cannot be ignored. The research community has dedicated significant efforts to this topic during recent years and a vast number of theoretical results have been published addressing different variants of the related mathematical problems. However, a comprehensive survey focusing on the technical and analytical aspects of the placement problem in various edge architectures is still missing. This survey provides a comprehensive summary and a structured taxonomy of the vast research on placement of computational entities in emerging edge infrastructures. Following the given taxonomy, the research papers are analyzed and categorized according to several dimensions, such as the capabilities of the underlying platforms, the structure of the supported services, the problem formulation, the applied mathematical methods, the objectives and constraints incorporated in the optimization problems, and the complexity of the proposed methods. We summarize the gained insights and important lessons learned, and finally, we reveal some important research gaps in the current literature.**

***Index Terms*—Edge/fog computing, MEC, cloudlets, resource orchestration, function placement optimization, offloading.**

I. Introduction

CLOUD computing has written a plenty of success stories for the last two decades. The amazing technological evolution together with the solid theoretical background established by the research community enabled several new applications and services to run in extremely large scale on top of different cloud platforms. Either public cloud platforms, such as Amazon Web Services [1], Google Cloud Platform [2], Microsoft Azure [3], or private ones based on open-source technologies, such as OpenStack [4], Docker [5] or Kubernetes [6], are capable of providing an "arbitrary" amount of virtual resources on demand by using recent virtualization techniques and resource management mechanisms. Well designed data centers encompass all the necessary physical assets, including thousands of blade servers and network devices, while the burden of operational tasks is delegated to the cloud providers.

However, during the last decade, the centrally placed physical resources started to move closer to the users in order to enable the operation of novel types of applications, such as latency sensitive ones. This paradigm shift has opened the door for telecommunications operators, mobile and fixed network vendors: they have joined the ecosystem to be part of the success story. Various concepts and paradigms appeared to designate the proper way how to leverage computing resources deployed in the vicinity of customers and end devices. Edge computing, fog computing, Multi-Access (formerly called as Mobile) Edge Computing, cloudlets are distinct concepts, nevertheless they share several common objectives and features [7]–[13]. Different technological and business use cases are addressed by these concepts but the telecommunications stakeholders are crucial players in all scenarios as standalone entities or federated with cloud providers. This paper does not attempt to excavate the precise distinction among these approaches, rather we focus on a key aspect central to all paradigms, i.e., placement methods.

We assume that the central cloud is extended with edge resources providing execution environments close to the users in terms of latency, e.g., in mobile base stations. By these means, customers' devices can offload computational tasks to this environment instead of consuming their local resources. Latency critical functions can also be offloaded from central clouds to the edge, enabling, e.g., critical machine type communication or real-time applications with strict delay bounds. As a result, novel types of services and distributed applications can be realized on top of a novel platform which tightly integrates network and cloud domains. Tactile Internet, remote surgery, augmented/virtual/mixed reality (AR/VR/MR) applications, future manufacturing based on Industry 4.0 are just highlighted examples which can reshape our digital society.

Manuscript received October 1, 2020; revised March 18, 2021 and July 24, 2021; accepted July 24, 2021. Date of publication July 30, 2021; date of current version December 8, 2021. This work was supported by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund through 2018-2.1.17-TÉT-KR, FK_20, and 2019-2.1.13-TÉT-IN Funding Schemes under Project 2018-2.1.17-TÉT-KR-2018-00012, Project 135074, and Project 2019-2.1.13-TÉT-IN-2020-00021. *(Corresponding author: Balázs Sonkoly.)*

The authors are with the MTA-BME Network Softwarization Research Group, Department of Telecommunications and Media Informatics, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, 1117 Budapest, Hungary (e-mail: sonkoly@tmit.bme.hu).

Digital Object Identifier 10.1109/COMST.2021.3101460

In addition, the emerging cloud platforms and the exposed capabilities have transformed the *software* running atop and also changed the corresponding software development techniques. Starting from monolithic applications running in dedicated virtual machines (VMs), microservices have emerged: consisting of loosely coupled, inter-communicating software modules running in separate containers or as distinct functions managed by the underlying cloud system. At the end of the day, developers and service providers can obviously benefit from this shift, however, several new challenges arise on the platform side.

A key problem, i.e., the optimal placement of *computational units*, stems from the geographically distributed nature of the virtualized infrastructure. Computational units can be VMs, software containers, tasks or functions as well, and the optimal hosting environment for them should be selected based on multiple criteria. The scheduler or the resource orchestrator is in charge of making that decision. Traditional cloud infrastructures are based on carefully designed data centers where the resources (compute, storage, network) are accommodated close to each other at central premises. The customized network topology of data centers provides extreme high bisection bandwidth. To put it simply, we have virtually zero delay and infinite throughput between the servers. Therefore the scheduler algorithms widely used in data centers cannot be invoked for novel edge systems as they do not take the wide-area network characteristics into consideration. Hence, novel mechanisms are required to cope with the new challenges of distributed cloud architectures where delays cannot be ignored. Similar techniques can be invoked by telecommunications operators to construct their own compound network services on demand over distributed edge environments.

The mathematical problem under the hood is generally a multi-criteria decision problem, where the optimal placement option must be selected for each computational unit. Specifically, multiple virtual compute infrastructures with different capacities and characteristics are available for hosting the computational units, and those entities should be able to communicate with each other with predefined quality dictated by the applications. The selection criteria include, but are not limited to the operating cost of the VM, the available computing and network capacity, hardware specific requirements (e.g., GPU is needed by the VM), the network delay and the available bandwidth between the VM and the end device, the battery status of the end device, the latency and the available bandwidth between the VM and another already deployed service component. Obviously, VMs, software containers or functions have to be instantiated differently and the network configuration also requires various methods, however, the formal mathematical problem of placement is similar in all cases. The careful placement of the computational building blocks may result in significant benefits in terms of application QoS, resource provisioning costs, or both.

Therefore, the research community has dedicated significant efforts to this topic during recent years and a large number of theoretical results have been published addressing different aspects and variants of the related mathematical problems. Various techniques of several scientific fields were applied from mathematical programming across graph theory to machine learning. In this paper, we provide a comprehensive survey focusing on the placement problem in the edge, which helps to categorize the proposed solutions and defines an adequate taxonomy to get a better understanding on the current status and on remaining research gaps.

A. Scope of This Survey

The scope of the survey involves all research results on the *optimal placement* of any type of computational units *over multiple options* of virtual compute infrastructure that differ in the provided resources and/or in attributes *that affect the QoS* of the deployed application. The emphasis is on the three terms in italic: optimal placement over multiple options that affect the QoS. Our focus is on the main technical and mathematical aspects of placement of computational units. We consider this architecture agnostic task as a core problem with significant importance affecting both the user experience and the related operation costs.

There are closely related and pertinent works that are worth mentioning in order to clearly define what falls out of the scope of our survey. Research results on deploying infrastructure elements instead of placing services on existing infrastructure are considered off-topic. Therefore, papers proposing analytical approaches for identifying the most attractive locations to install edge nodes [14], [15], fog nodes [16], [17] or data centers [18] based on impacting factors and key challenges to reduce the costs associated with their deployment and maintenance, or to support the requirements of mobile and latency-sensitive applications are not in our focus. Furthermore, papers that describe and solve cloud scheduling problems within a single data center, e.g., [19], are also omitted from this survey. Even if the scope of the research involves multiple data centers or edge clouds, we do not cover those papers that do not devise any optimization problem out of the placement challenge over edge and/or cloud resources. Hence, papers that focus on a framework description without in-depth algorithmic analysis [20]–[23] are not included.

Research studies that propose placement solutions of computational units whose effectiveness depend on the result of another prior allocation scheme are also considered off-topic. Hence, e.g., papers from the field of Coded Distributed Computing (CDC) [23] are not included, since the computation function allocation is impracticable without the prior placement of the files to be processed.

We have found related works that comply with the condition of containing a formal optimization problem, tackling it with an algorithmic approach, although they do not formulate a placement decision involving more than one edge node [24]–[28], or do not interpret the offloading problem in its widely understood form [29]–[31]. We argue that offloading problems that are defined as binary decisions [32]–[35], i.e., to offload or not to offload, should not be compared to research results that face a significantly larger decision space, therefore we exclude those research efforts as well. These optimization

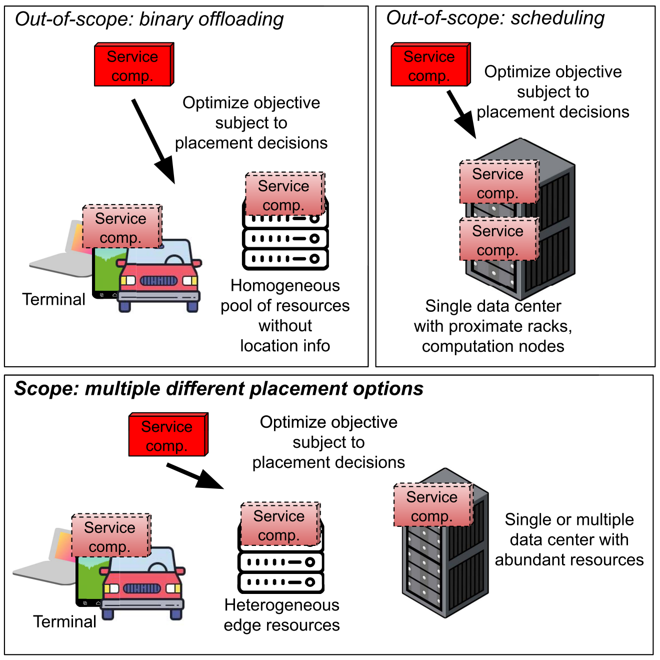

Out-of-scope: binary offloading
Service comp.
Optimize objective subject to placement decisions
Service comp.
Service comp.
Terminal
Homogeneous pool of resources without location info
Out-of-scope: scheduling
Service comp.
Optimize objective subject to placement decisions
Service comp.
Service comp.
Single data center with proximate racks, computation nodes
Scope: multiple different placement options
Service comp.
Optimize objective subject to placement decisions
Service comp.
Service comp.
Service comp.
Single or multiple data center with abundant resources
Terminal
Heterogeneous edge resources

Fig. 1. Illustration of the optimization problem considered in this survey, and the problems considered in out-of-scope research areas.

problems and their comparison to our considered scope are illustrated in Figure 1.

The decision of omitting these related papers serves the goal of making this survey comprehensive in the selected topic. First, by adding the excluded cloud scheduling and binary offloading papers, a comprehensive survey would be intractable due to the sheer amount of related papers. Second, although those work that fall in the aforementioned domains and contain algorithmic solutions might seem, at a first glance, similar to the selected papers, their optimization goals and constraints are inherently and significantly different. Therefore, the insightful comparison we provide in the next sections could have been impossible for such research results. Third, although we consider the practical implications and outcomes of academic research very important, we place the primary focus on mathematical modeling and problem solving in this survey, rather than on architectural design and implementation achievements.

B. Contribution

This survey provides a comprehensive summary on the mathematical problem of placement of computational units in emerging edge infrastructures operated by either mobile/fixed network operators or cloud providers or a federation of them. This article surveys the related literature over the period 2015-2020. The main contributions are the following.

- First, we define a **hierarchical taxonomy** suitable for the classification, categorization and understanding of placement methods proposed by researchers in the revolutionary age of edge computing.
- Second, the surveyed papers are **analyzed and categorized** according to the dimensions of our taxonomy. The main features, capabilities and limitations of different approaches and mathematical tool sets are gathered and summarized, which might be useful in product development by mobile or fixed network vendors, operators or cloud providers.
- Third, important **research gaps** and future research directions are revealed. Those that formulate theoretical problems are relevant to academic researchers, others raise practical issues for network vendors, operators and cloud providers.
- Finally, the gained insights and important **lessons learned** are summarized as a result of a thorough comparison and investigation.

```
taxonomy                                    Section
 |_platform                                      IV
 |  |_components                               IV-A
 |  |  |_cloud-edge                           IV-A1
 |  |  |_multi-cloud                          IV-A2
 |  |  |_multi-edge                           IV-A3
 |  |  |_edge-terminal                        IV-A4
 |  |  |_cloud-edge-terminal                  IV-A5
 |  |_policy                                   IV-B
 |  |_requests                                 IV-C
 |_application                                    V
 |  |_type
 |  |_security & privacy
 |  |_structure
 |     |_single component                       V-A
 |     |  |_offloading                         V-A1
 |     |  |_edge/fog computing                 V-A2
 |     |_multiple components                    V-B
 |     |  |_offloading                         V-B1
 |     |  |_edge/fog computing                 V-B2
 |     |_multiple connected components V-C
 |        |_offloading                         V-C1
 |        |_multi-cloud                        V-C2
 |        |_edge/fog computing                 V-C3
 |_mathematical optimization                     VI
    |_problem formulation                      VI-A
    |_applied methods                          VI-B
    |_objectives                               VI-C
    |_constraints                              VI-D
    |_complexity                               VI-E
```

Fig. 2. Our proposed taxonomy.

The structure of our proposed taxonomy is shown in Figure 2. We have identified three top level dimensions that determine the placement problem and the methods that can be applied to solve the problem. Namely, *i*) the platform capabilities including the cloud-edge architecture, the placement controller policy and the placement request types (offline vs. online processing), *ii*) the application characteristics encompassing the types of supported applications, the structure of the services (single vs. multiple components, isolated vs. interconnected components), and the supported level of security & privacy, and finally *iii*) the mathematical aspects covering the problem formulation, the applied optimization methods, the objectives and constraints of the optimization models and the complexity of the proposed solutions.

The outcomes of the analysis and the detailed insights can be beneficial to researchers in multiple ways. First, the collected and summarized mathematical tool set provides a good starting point for related problems in other research fields. The pros/cons of different approaches can be identified in advance based on the lessons learned and suitable methods can be selected from a restricted search space for further investigation. By these means, the design time of the algorithms for emerging but related problems are shortened and the pointers to the technical details of promising, exploitable solutions are available. Second, this survey delivers a comprehensive catalog on the mathematical apparatus for the placement problem, which can be beneficial for applied research and product development. For example, this catalog can foster the design and implementation of an orchestrator software of an arbitrary edge cloud platform. For relevant solutions, the complexity characteristics are also revealed which helps to assess the feasibility of the approach in the targeted environment. Third, the revealed research gaps outline promising future research directions which could trigger dedicated activities on challenging topics.

C. Outline

The rest of the paper is organized as follows. In Section II, the background is introduced via the state-of-the-art of edge computing. In Section III, we provide a detailed description on the taxonomy that we follow in the paper. Section IV presents our analysis according to the first dimension of our taxonomy, focusing on the characteristics of the cloud platform. Section V is devoted to the second dimension addressing the application related features and properties. The third dimension targeting the mathematical aspects is investigated in Section VI: in Section VI-A, the papers are grouped according to the problem formulation, while in Section VI-B we analyze the papers in terms of the applied mathematical methods; in Section VI-C and Section VI-D, we review the optimization goals and constraints used in the collected papers, respectively; Section VI-E describes the complexity aspects of the proposed algorithms in the reviewed papers. Section VII highlights the revealed research gaps and potential future research directions, while Section VIII provides a summary on the lessons learned and concludes the paper.

II. STATE-OF-THE-ART OF EDGE COMPUTING

We devote this section to describing the background of the domain that our survey covers. To this end, we briefly present other surveys that have partly touched upon the topic in our focus. Together with the descriptions, we make a clear separation from those works to clarify the exact scope of the present survey. For easier tractability, we provide a summary of acronyms used throughout the paper in Table I.

TABLE I
SUMMARY OF IMPORTANT ACRONYMS USED IN THIS PAPER

Acronym	Definition
4G	Fourth-Generation Mobile Telecommunications Technology
5G	Fifth-Generation Mobile Telecommunications Technology
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
AWS	Amazon Web Services
BS	Base Station
BW	Bandwidth
CC	Cloud Computing
CDC	Coded Distributed Computing
CPS	Cyber-Physical System
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
E2E	End-to-End
EH	Energy Harvesting
ETSI	European Telecommunications Standards Institute
FaaS	Function-as-a-Service
FDMA	Frequency-Division Multiple Access
GA	Genetic Algorithm
GPU	Graphics Processing Unit
HDFS	Hadoop Distributed File System
ILP	Integer Linear Program
INLP	Integer Nonlinear Program
IoT	Internet of Things
LP	Linear Program
LSTM	Long Short-Term Memory
MANO	Management and Orchestration
MCC	Mobile Cloud Computing
MEC	Multi-Access Edge Computing / Mobile Edge Computing
MILP	Mixed-Integer Linear Program
MINLP	Mixed-Integer Nonlinear Program
MIQCP	Mixed-Integer Quadratically Constrained Program
ML	Machine Learning
MR	Mixed Reality
NF	Network Function
NFV	Network Function Virtualization
NP	Non-Polynomial
QCQP	Quadratically Constrained Quadratic Program
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RG	Resource Graph
SA	Simulated Annealing
SAP	Service Access Point
SAT	Boolean Satisfiability Problem
SC	Service Chain
SDN	Software-Defined Networking
SFC	Service Function Chain
SGE	Service Graph Embedding
SINR	Signal-to-Interference-plus-Noise Ratio
SLA	Service Level Agreement
SNR	Signal-to-Noise Ratio
UAV	Unmanned Aerial Vehicle
UE	User Equipment
VM	Virtual Machine
VNE	Virtual Network Embedding
VNF	Virtual Network Function
VR	Virtual Reality
WiFi	Wireless Fidelity
WPT	Wireless Power Transfer

Traditional Mobile Cloud Computing (MCC) combines cloud computing and mobile computing. As a result, the computational capacity of the mobile devices is augmented making use of different types of cloud resources. The original MCC concept assumes distant data centers enriching available mobile services, however, its generalized form encompasses

TABLE II
CATEGORIES OF THE REFERRED SURVEY PAPERS IN TERMS OF THE COVERED COMPUTING PARADIGM

Computing paradigm / **Covered area**	Multi-Access Edge Computing	Edge/Fog Computing	Mobile Cloud Computing	NFV/SDN
Holistic approach to the topic	[7]–[9], [36]–[38]	[10]–[13]	[39], [40]	[41], [42]
Focused on placement problems	[43]–[45]	[46]	[47]	[48]

other types of cloud elements, such as proximate data centers (e.g., cloudlets) and mobile computing entities with shared resources. The surveys in [39], [40] summarize and categorize the efforts of executing mobile applications external to the mobile device in the concept of MCC. Although, the papers focus on leveraging cloud resources for single computation offloading, they also shed light on complex offloading decision problems over heterogeneous resources including proximate cloudlets and multi-tenant clouds.

Multi-Access Edge Computing (MEC), originally known as Mobile Edge Computing, is an evolved version of MCC and it is also a well-studied computing paradigm where the focus is on providing computation capabilities within the radio access network (RAN) in close proximity to mobile users. In the literature, the borders are not sharp and sometimes MEC features are also implied in the MCC concept. One of the most recent and extensive study on leveraging distributed resources at the mobile network edge is the work in [7]. The survey covers a comparison of recently emerged computation paradigms for delay sensitive and context-aware Internet-of-Things (IoT) services, as well as a state-of-the-art research related to "end-edge-cloud" orchestrated networks with special attention to computation offloading, caching, security, and privacy. Another recent study [36] provides a holistic overview of the MEC technology and its potential use cases in terms of integration with 5G enabler technologies. The authors also discuss the applicability of many leading-edge technologies with MEC, such as wireless power transfer (WPT) and energy harvesting (EH), Unmanned Aerial Vehicles (UAVs) and machine learning (ML), in great details. Another review on the exploitation of the MEC paradigm by the realization of IoT applications is presented in [37], in which the emphasis is placed on the technical aspects of MEC-IoT synergy and on other integration technologies for 5G, mentioning service deployment/resource allocation in coarse details. The survey in [38] also presents a holistic overview on the MEC-IoT interaction with increased focus on the performance of the different edge network architectures and IoT reference applications in terms of latency, bandwidth and energy consumption, along with the potential security challenges. Other comprehensive surveys summarizing and categorizing the efforts on MEC from an overarching point of view are published in [8] and [9], where the main focus is on the research issues related to the joint radio and computation resource allocation, including different computation offloading and mobility management scenarios. Both papers discuss the advancements in the process of MEC standardization, as well as defining and examining reference applications and their major use cases.

Similarly to MEC, fog computing has also gained significant attention over the recent years as an alternative approach to the centralized cloud computing model, leveraging fog node resources at the edge of the network along with the management of connected communication resources. The authors in [10] compile a comprehensive survey of recent efforts about fog-enabled architectures in the context of IoT applications, including the major similarities and differences compared to other computing paradigms. They provide a large-scale and thorough overlook on the state-of-the-art publications, as well as designating the open research challenges and future directions, partially reviewing placement-oriented articles. There are other surveys [11]–[13] that also discuss the concept of fog computing in the light of infrastructural, Quality of Service (QoS), Quality of Experience (QoE), resource management and allocation challenges, mostly about architectural and non-placement decision problems.

Compared to these aforementioned surveys, in our work we cover all architectural solutions that have been proposed in the literature for deploying services in the Cloud Continuum, and we steer our focus to the technical and analytical aspects of function placement. In this sense, the current survey is broader in scope regarding the architectures than any of the related surveys, and digs deeper in the analysis of service orchestration, which we consider to be one the main technical challenges of such platforms.

Besides the aforementioned holistic surveys of edge-related computing paradigms, there are several publications which investigate and review optimization problems related to edge resource utilization in more details, typically from the perspective of either the end devices or that of the central/remote cloud. Based on recent works in edge computing, the authors of [48] examine and classify the existing solutions for virtual machine and Virtual Network Function (VNF) placement based on their static/dynamic nature and performance metrics. Although, the survey thoroughly analyzes the recent algorithmic approaches of cloud-based multi-component service placement, the research challenges related to edge resource management and their solution techniques are only addressed partially. The recent study of [46] defines a taxonomy for the categorization of architectures related to edge resource management in terms of resource type, management objective, location and utilization purposes. While the study discusses many articles in a wide range from edge resource discovery through resource allocation up to resource sharing, it leaves out several important placement-specific aspects including, e.g., cloud-to-edge offloading scenarios. Comparably, the related survey of [43] presents a comprehensive review on MEC-related orchestration complemented with the analysis of the MEC reference architecture, standardization activities and open research challenges. The paper provides an overview on specific orchestration scenarios considering individual services and an edge-cloud platform network, but other edge-related service deployment approaches are not addressed.

A few surveys summarize the field of MEC-related resource optimization from a different perspective. The authors of [44] state the importance of machine learning algorithms in MEC scenarios, where the massive number of end devices, varying characteristics of applications and user mobility exceedingly increase the dimensionality of task offloading and resource allocation problems. The survey offers an insight into machine learning solutions for MEC systems, excluding other relevant optimization techniques. In [45], the authors investigate the application of game theory techniques on the major challenges imposed by MEC services, in which several game models are considered for resource constrained optimization problems. The survey in [47] explores the challenges and algorithmic methods of multi-objective decision-making for time-, and energy-aware task offloading, however, it mainly takes the MCC paradigm into account and refers to MEC only in the context of hybrid decision-making.

Away from the edge, the authors of [41] provide a comprehensive survey on the advancements of Network Function Virtualization (NFV) with an extensive and in-depth discussion on VNF algorithms. They examine relevant use cases, including VNF placement, scheduling, and migration, but omit the analysis of VNF-based service placement problem considering resources at the edge. In their extensive survey of [42], researchers analyze and classify resource provisioning algorithms based on their mathematical formalization, optimization objectives, constraints and efficiency. They discuss topics related to our focus, such as VM migration and multi-cloud scenarios, however, the survey focuses only on the long-established cloud computing (CC) paradigm.

Many of the aforementioned surveys, which are categorized in Table II, investigate algorithmic problems and their literature in connection with edge resource allocation, such as task offloading, content-caching, VM migration and server partitioning, but none of them address explicitly the challenges regarding placement problems in the edge, in which a number of computation nodes are available at the edge of the network. This is one of the distinguishing features that characterizes our survey.

III. Taxonomy

We provide a taxonomy concerning the most important aspects of research on placement in clouds. We create three main groups of dimensions that we later use to characterize the body of work. The groups of dimensions determine the structure of the comprehensive view that we provide: in Sections IV, V and VI, the presented analysis of the related work follows the same grouping. The dimension groups touch upon the following aspects in this order: *i*) cloud/edge platform features, *ii*) application-specific details, *iii*) mathematical modeling of the placement problem. In this section we summarize the dimensions that constitute the three groups.

A. Platform characteristics

Within the group of cloud/edge platform-related characteristics, we define the following three dimensions: *i*) platform components, i.e., what type of infrastructure elements are assumed, *ii*) the entity responsible for placement logic, i.e., which platform element decides where to place the service components, *iii*) placement request processing, i.e., either batch or online processing. We provide an in-depth discussion of the options of all these dimensions in Section IV with the respective classification of the collected papers.

Platform components: The collected research papers differ in the set of layers they focus on in terms of potential platforms for the placement of computational units. We illustrate the whole set of layers in Figure 3. Some components and features are related to the cloud providers while others are under the control of the network operators (either mobile or fixed ones).

In the top layer we consider the clouds, in plural to account for the multi-cloud setting, i.e., a number of data centers. In general, those are public clouds, assuming quasi-infinite compute resources, to be leased for a relatively low price, while a high level of availability is guaranteed by strict service level agreements (SLAs). Besides the three giants providing the leading public platforms, i.e., Amazon Web Services [1], Google Cloud Platform [2], Microsoft Azure [3], other stakeholders have also entered the market. Typically, these platforms expose only restricted APIs to control the placement decisions. For example, the larger geographic regions or availability zones can be defined for the applications to be deployed. However, the connections between the regions are typically out-of-scope for the cloud providers which is a severe limitation regarding the supported applications. A straightforward solution is to establish a dedicated business collaboration among the cloud providers and the involved network operators. In multi-cloud scenarios, a higher level orchestrator makes the placement decisions, selects the appropriate cloud domains from the available ones and triggers the deployment via the corresponding APIs. In this case, the business relations among the participating stakeholders can be quite complex and new business models and pricing schemes are required. On the other hand, private cloud platforms built from open source components, such as OpenStack [4], Docker [5] or Kubernetes [6], enable finer granularity in the placement control. Moreover, the available APIs and the underlying capabilities can easily be extended and tailor-made features, policies and custom algorithms can be implemented. In addition, public and private solutions are mixed in hybrid platforms posing new challenges to the placement, especially in terms of security and privacy.

In the middle layer there is the edge, generally either at base stations (BS) and central offices of mobile networks, i.e., mobile edge, or at dispersed locations of a wired network's infrastructure. In any case, the edge nodes are assumed to be limited in compute power, expensive to operate and maintain, prone to errors and downtimes. On the other hand, the edge computing infrastructure offers low latency for the end users due to its physical proximity. The "edge realm" is the key area for network operators to enter the ecosystem because they have the physical footprint close to the potential customers. The ownership of the edge infrastructure has important impact on the placement control, as well. The edge resources can be owned either by the cloud provider (managing the central cloud resources as well) or by other operators (e.g., mobile

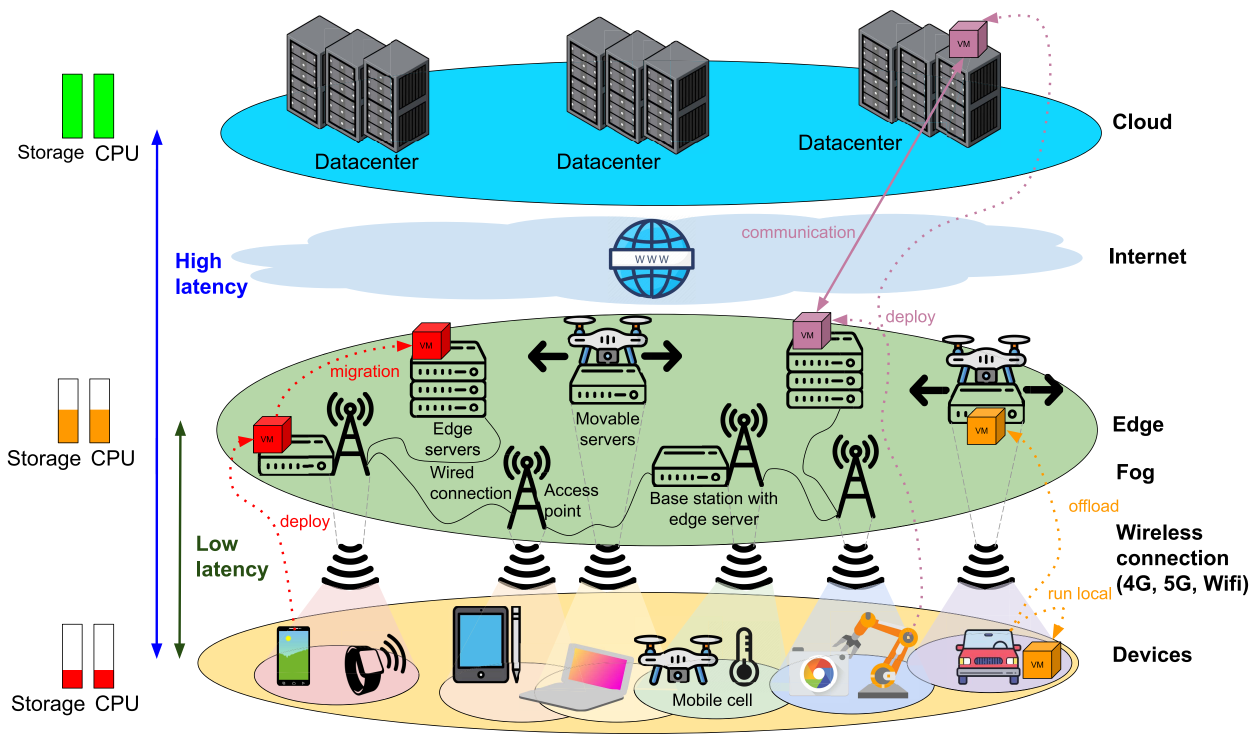


Fig. 3. Architecture and the features of a distributed computing system.

network operators) or even by the customers. Furthermore, the concerned network infrastructure plays a crucial role in the quality of the provisioned services. It can either belong to the network operator owning the edge resources or dedicated connectivity services with configurable quality parameters are provided for the edge provider in order to be able to control the quality of the connections. When all resources are managed from a single administration domain, a central entity can be in charge of orchestrating both central cloud and edge resources, and also the networks. For example, in case of OpenStack, edge servers can be added as distinct compute nodes, however, the default orchestration service has to be extended in order to be aware of the underlying network topology and network characteristics. If the cloud and edge resources are owned and managed by different stakeholders and network operators are also involved, more complex mechanisms and workflows are required. For example, a dedicated orchestrator is needed on the top to calculate the optimal placement and to enforce the appropriate deployment of service components and network configuration making use of lower level orchestrators and exposed APIs. As these interfaces are mainly technology dependent ones, it is worth noting that the implementation could be cumbersome, especially when different technologies have to be integrated.

At the bottom layer we have the myriad of terminals that include mobile user equipment (UE), i.e., cellphones, industrial robots, self-driving cars, smart buildings, IoT devices, etc. These terminals are considered to be the targeted users of the applications, of which the operations are optimized with the placement methods proposed in the collected research papers. Terminals are assumed to have limited power, usually running on battery, and they are considered to be poor in terms of computation capacity. The access network type is out-of-scope in this survey, although the collected papers predominantly assume wireless access to the applications deployed in the cloud/edge platforms, e.g., MEC applications. While telco NFV use cases operate over wired networks, the end users of those are in the end accessing the service via mobile networks.

Location of placement logic: When the overall computing infrastructure contains only cloud and edge layers, then the question of authority does not arise: the silent assumption is made that the cloud (or edge) operator orchestrates its own technological domain and also manages the involved network infrastructure. However, if a research work considers the possibility of running computation on terminals as well, then the task placement policy may be dictated by one of two parties: either by the cloud (or edge) operator, or by the owner of the terminal. This ambiguity is also reflected in the term of offloading that is used sometimes in terms of delegating tasks from terminals to the edge, sometimes vice versa.

Processing placement requests: Similarly to data processing big data engines, in the field of placement methods we can make a differentiation in regard to the way placement requests are processed. There are solutions proposed for the batch processing of all the requests at once, called offline methods. The other large group of approaches is that of the streaming-like methods, which process each and every incoming request individually. We call this group as online methods, and we distinguish those that apply migration of computation units that have been already placed if the placement of the actual request deems it necessary, e.g., in order to free up resources. The online placement algorithms are aware of the current (and past) state of the system in terms of resources, but they have no certain knowledge of the future, e.g., future service requests or failures. Therefore, in most cases allocations are made without any assumption of resource requests that arrive in the future. In the intersection of the two groups, there are the hybrid methods, which are adaptive solutions that place the components, i.e., tasks, VMs, containers, etc., into the edge and/or the cloud infrastructure in an offline manner, but migrate them dynamically if required. The continuously changing conditions, e.g., the mobility of end devices or edge nodes, trigger the online changes in the placement for achieving various goals, e.g., to follow mobile users, to maximize the coverage, or to minimize the processing time of the offloaded tasks.

B. Application Related Aspects

The second group of distinctive dimensions contains application related aspects. Our taxonomy defines three dimensions here: *i*) the use case of the application, *ii*) application structure, i.e., how many computation entities are built from the application's code implementation, *iii*) security and privacy features. These dimensions seem to be independent from one another (and also from those in the other dimension groups), but as we delve into the details in Section V, we show that certain patterns can be observed.

Application type: The emergence of edge and fog computing has been partially driven by specific use cases, along with the appearance of novel applications. In the research papers related to our survey scope these applications are often specifically determined. If the application type is not emphasized, the researchers assume general applications and the proposed solutions are application agnostic to some extent. We argue that the application type for which service components must be appropriately placed in a cloud/edge infrastructure bears great importance. Consequently, we make a separation of the processed research papers along this dimension later in Section V. The main application types are Internet of Things, Industry 4.0, big data analytics, and telco (or telco-related) NFV.

Internet of Things (*IoT*) applications encompass a vast number of IoT devices, such as various sensors generating a massive amount of data or smart devices taking part in bidirectional communications, and different types of processing units capturing, pre-processing the data and also implementing the main business logic. In a typical setup, the IoT sensors provide continuous data streams which are eventually processed by remote applications in a consistent way. Usually a dedicated IoT gateway bridges the communication gap between the devices and the cloud/edge domain: it is in charge of pre-processing, aggregating and filtering the data, and also routing the traffic towards the next level processing entities. The gateway can include hardware related elements managing the communication, but the other aforementioned functionalities can be implemented in software and the corresponding components could be orchestrated similarly to other processing functions. Typically, the gateway modules need to be placed in proximate hosting nodes with limited processing, storage, battery and bandwidth capacity which should be taken into account in the placement decision. In addition, the IoT devices and the characteristics of the generated data vary over a wide range. For example, a data stream from an environmental monitoring system or from a smart plant application significantly differs from a video stream which is used by, e.g., an online face recognition or object detection application. Both the sending (or retrieving) frequency and the amount of the transferred data vary yielding different requirements on the underlying network and compute resources. The application itself also implies specific requirements. For example, critical IoT applications, such as autonomous vehicles, require strict and low response latency which affects the placement of the processing units. In general, the placement algorithms are expected to be capable of *i*) resolving arbitrary latency constraints defined between given components of the IoT application or end-to-end latency bounds, *ii*) taking different bandwidth/throughput profiles into account, *iii*) optimizing energy consumption to prolong battery life of the devices and the IoT gateway (by these means optimizing operational costs as well), and finally, iv) dynamic mapping and placement which is required in case of mobility support (e.g., intelligent transportation) when the IoT nodes are in constant motion.

Industry 4.0 is a name for the current trend of automation and data exchange in manufacturing technologies and it also includes a special subset of IoT, referred to as Industrial IoT (IIoT). Industry 4.0 applications pose strict QoS requirements in order to guarantee real-time operation and to ensure time-optimized service delivery. The placement task is challenging due to the diversified data sensing frequency of different industrial IoT devices and their reported data size. For example, robot or UAV navigation applications are essential components of future manufacturing environments which require sophisticated image processing based on live video streams. The control commands must arrive with strict timing at the robots therefore the careful selection of the execution environments (running the controller codes) and the network paths (conveying the commands) is crucial. These mission and safety-critical applications require additional features to be offered by the orchestration system. For example, dependability and reliability can be guaranteed by duplicated controller instances (replicas) deployed to physically different servers which are connected to the controlled plant via disjoint paths. The theoretical constraints under the hood are generally referred to as anti-affinity rules which are respected by the embedding algorithms.

Big data analytics is another relevant application type. As the amount of data collected in various IT systems has grown exponentially in the recent years, capturing, storing, processing, querying, updating and analyzing data while fulfilling strict time criteria and effective resource consumption are challenging tasks. Big data platforms provide versatile solutions typically designed for single data centers. However, big data applications can also benefit from recent edge/fog computing technologies. For example, data processing performed at the edge can provide faster actuation and can also reduce the network load. In a hybrid edge/cloud environment, where the application components consume compute and storage resources from the central cloud and edge infrastructures, the task (and data) placement algorithms have significant impact on the performance. On the one hand, generally a large amount of data is moved among the task executors which requires careful network path selection and proactive bandwidth allocation. On the other hand, sophisticated placement of the executors can largely reduce the overall network load and the end-to-end processing latency which is crucial for time-critical stream analytics often demanding real-time responses (e.g., within 100 ms in case of computer vision applications).

NFV based telco (and telco-related) services have been important drivers of the evolution of the MCC and MEC paradigms. Telco services (e.g., mobile multimedia applications) are generally provisioned for a very large costumer base while respecting strict SLAs, including availability, reliability, bandwidth, delay and mobility requirements. These network services are described and deployed in the form of Service Function Chains (SFCs), each consisting of an ordered set of VNFs. Certain requirements can be defined for the overall service (e.g., availability, reliability) while other ones target specific scopes (e.g., delay bound between two VNFs, minimum bandwidth for a given path including multiple VNFs). Other constraints related to dependability, such as anti-affinity and link anti-affinity rules, pose additional challenges to the orchestration system and the placement algorithms. Moreover, as telco VNFs serve thousands of customers, these VNFs are usually shared among multiple SFCs in order to save resources, improve utilization and reduce operational costs. The underlying infrastructures follow multi-tier hierarchy and encompass heterogeneous resources from different cloud and network domains. In order to enable the dynamic and flexible provisioning of compound telco services consisting of multiple constituent VNFs, telecommunications operators need sophisticated placement methods. These mechanisms are implemented (or more precisely, will be implemented) in the orchestrator products of different vendors. The ultimate optimization objectives from the operators' perspective are the operational cost and the revenue which drive the design (or the configuration) of the placement methods and the orchestration mechanisms. In case of telco services, the consumers are typically human end users whose behavior (e.g., mobility, daily profile) can be predicted and used as an input in the placement algorithms.

Service structure: After the dimensions that characterize the compute infrastructure and the placement logic, we define a dimension to describe the application to be placed, as well. From the perspective of placement, we argue that it is essential whether the application is monolithic, or it can be divided into components for which placement decisions might be made separately. In addition to those two categories, we make the distinction of the latter's sophisticated variants, which consider some type of dependence between the components, should it be latency budget, affinity (collocation), anti-affinity (requiring physically different underlying resources), or the like. Telco services typically fall into this latter group. In Section V we give a survey of the collected research works following these categories.

Security and privacy: For certain applications it is inevitable to consider security and privacy aspects as early as in the design phase of the orchestration platform. Several researchers have done so, and we decided to emphasize and acknowledge their effort, so we define the third dimension of this group based on the security and/or privacy aspects taken into account in the related papers. Unfortunately, only a small fraction of the related papers give these important aspects any consideration.

C. Mathematical Modeling

As stated in Section I-A, only those papers are considered in the scope of this survey that provide a mathematical model and an in-depth algorithmic analysis of the placement challenge. The third group therefore constitutes the evaluation criteria of the papers in terms of analysis and algorithmic design. Respectively, the dimensions in this group cover the *i*) formalized model, *ii*) approach to the problem, *iii*) optimization goal, *iv*) constraints that are taken into account, and *v*) complexity of the proposed algorithms. Detailed explanation, evaluation, and summarizing tables are given in Section VI on the covered papers along these dimensions.

Placement problem formalization: Beside the dimensions that characterize the infrastructure and the application composition, we also account for the mathematical tool set the researchers propose to apply. First and foremost, the formal model of the optimization problem is what characterizes the theoretical contribution of a research paper. We therefore discuss all the applied modeling frameworks in Section VI-A. As a foreword, we can state that the prevalent formalization technique is the family of linear programming, as the clear majority of papers apply some kind of variant of those for transforming the quest for optimal placement into the words of mathematics.

Problem solving approach: In addition, perhaps as the most interesting part for fellow researchers, we also describe the content of the collected papers by listing the approaches they chose to solve the optimization problems. This is the dimension that shows the largest heterogeneity over the related work, demonstrating that researchers have considered the placement problem interesting and thus made the effort to tackle it in innovative ways. In Section VI-B we present all the algorithms that we have seen in the related work.

Optimization objective(s): Essentially all the collected research papers strive to optimize some kind of notion with the careful placement decisions. Therefore, it is obvious that a categorization based on the optimization goal is of paramount importance. In Section VI-C we introduce those goals in details, here we just briefly list the most widely applied ones: delay, energy, revenue, resource utilization.

Optimization constraint(s): Along with the optimization come the constraints. Throughout the papers we studied, the variety of constraints considered is much richer than that of the optimization goals. We devote Section VI-D to discussing those; in general, many researchers take into account network and compute limitations, e.g., delay and bandwidth capacity for the former, available CPU cores and memory for the latter. Papers addressing MEC tend to form constraints on radio resources, e.g., radio channels, carriers, signal-to-noise ratio (SNR), and user mobility.

```
taxonomy
├─ platform
│  ├─ components
│  │  ├─ cloud-edge
│  │  ├─ multi-cloud
│  │  ├─ multi-edge
│  │  ├─ edge-terminal
│  │  └─ cloud-edge-terminal
│  ├─ policy
│  └─ requests
├─ application
│  └─ ...
└─ mathematical optimization
   └─ ...
```

Fig. 4. The taxonomy angle analyzed in Section IV.

Complexity analysis: Finally, we summarize the complexity characteristics of the proposed solutions in the corresponding papers. Several research works provide detailed complexity analysis on the presented algorithms and explicit formulas are derived. In Section VI-E, we introduce a common notation to make those results comparable to each other. In addition, some papers conclude only textual characterization and specify the final complexity to be polynomial or exponential, while only sub-problems or sub-steps of the proposed solutions are analyzed in details. Other works define the complexity of their heuristics in terms of the complexity of external algorithms, such as LP solvers.

IV. The Cloud Architecture

In this section, we start the characterization and categorization of the collected research papers along the first group of dimensions defined in Section III. Specifically, we draw the high level system architecture of edge cloud and fog systems in order to roughly position each of the collected papers regarding the type of infrastructure they address, and how the placement decisions are made therein. As a reminder, we indicate the taxonomical aspects studied in this section by bold fonts in Figure 4.

A. Platform Components

We first characterize each research paper by the position the published solution takes in the cloud-edge architecture that exhibits three major layers in our understanding. We depict an illustration of the overall infrastructure of virtual computation in Figure 3 with these layers. On top, we model the datacenters in the core of the Internet as the cloud layer. At the edge of the Internet, there are various types of edge computation nodes. Edge servers deployed at mobile base stations or any other wireless access points, edge servers close to the last mile of wired networks, even compute resources mounted on vehicles, like the movable servers built on UAVs. At the bottom, we find the end-user devices and terminals: mobile/cell phones, tablets, smart watches, laptops, sensors, IoT devices, industrial robots, connected or autonomous vehicles, etc. One common attribute of these devices is that they have a demand for computation resources: either by generating important data, or by requiring the results of some computation to actuate upon or just visualizing those. The other characteristic these devices share is that in most cases they are connected by wireless technology to the cloud-edge infrastructure.

We make five distinct categories for the catalogue of collected papers on the combination of the three layers. These are the following: *i*) central cloud and edge nodes are considered for task placement, *ii*) only the cloud, but numerous data centers offer placement options, *iii*) similarly, only edge sites are assumed, but many of them, iv) edge nodes are available for offloading some of the tasks from devices, v) both central cloud and edge servers can be selected for offloading. We briefly discuss all these options.

1) Cloud-Edge: When papers consider the two-layer infrastructure of central cloud, i.e., a data center with abundant resources but at a remote location from the perspective of the users, and edge computing nodes at the edge of the network close to the users, although low on computational resources and expensive to operate, then the primary question is usually to move delay-sensitive operations from the cloud to the edge. Vice versa, papers might approach the same placement option by moving computation from the edge to the cheaper cloud for tasks that are indifferent to the latency this step adds to the application's quality of experience. The opposing effects that create the optimization problem are often costs and delay, i.e., it is considered to be less costly to operate a remote data center than to operate the same amount of computational capacities in edge nodes, however this comes at the price of increased end-to-end latency between the end user and the cloud.

2) Multi-Cloud: Although, intra-cloud scheduling is out-of-scope, we argue that inter-cloud task placement, possibly in a multi-operator setting, leads to algorithmic problems that are similar and thus comparable with those in the cloud-edge setting. Therefore, we decided to keep those papers that attack multi-cloud placement problems in scope. We list the papers into this category that primarily do not leverage on any infrastructure options in the edge and/or on the devices. In case they do, the distinctive characteristics being edge nodes and end devices are not taken into account in the problem statement. In this category, usually the research problem is induced by the difference in cloud service offerings: price, SLA, VM flavors.

3) Multi-Edge: We list the papers into this category that address placement options solely on edge nodes, i.e., no cloud or end devices are assumed to be available. Similar to the multi-cloud category in the sense that all placement options fall in one layer in our architecture view, this category however inherently differs in the reasons that lead to the placement problem. The most frequent reason is the heterogeneity of application latency requirements: edge nodes are dispersed geographically, therefore provide different delay characteristics from a given user. On the other hand, applications' delay requirements are also heterogeneous, which creates a complex setting when tasks are to be mapped to edge servers, exacerbated by the often made assumption on limited resources in edge nodes.

4) Edge-Terminal: We create a category for the traditional MEC offloading papers in which user devices delegate

TABLE III
CATEGORIES IN TERMS OF PLATFORM COMPONENTS, OWNER OF PLACEMENT LOGIC AND TEMPORAL PLACEMENT DECISION-MAKING

Platform components	Multi-cloud	Cloud-edge	Multi-edge	Cloud-edge-terminal		Edge-terminal	
Policy logic				in cloud platform	in terminal	in edge platform	in terminal
Offline	[49]–[53]	[54]–[77]	[78]–[98]	[99]–[107]		[108]–[116]	[117]
Online	[52], [118]–[124]	[59], [125]–[128]	[129]–[140]	[100], [141]–[146]	[141], [143], [147]–[153]	[109], [111], [154]–[160]	[113]
Online with migration		[161]–[167]	[168]–[170]				
Hybrid	[171], [172]	[173]–[179]	[180]–[182]				[183]

processing tasks to the edge computing infrastructure, usually for battery conserving purposes. In the papers that belong to this class there is no cloud option to offload tasks to. On the other hand, there are multiple options of edge nodes in the collected papers, because we omit research results on binary offloading decisions. Usually the challenges addressed in these papers stem from the choice of edge nodes, the latency requirements, and compute capabilities.

5) Cloud-Edge-Terminal: Finally, the last category might be considered as an extension of the previous one by adding remote clouds as options to the offloading decision. Naturally, only those computational tasks can be offloaded to the central cloud that are less sensitive to delay, so it is beneficial to offload them to the cloud, instead of consuming precious edge resources.

B. Location of Placement Logic

In Table III, we sort the collected papers into the aforementioned five categories, represented in columns of the table. Within those columns, for the MEC offloading categories we further distinguish papers based on the placement decision maker the papers assume: whether the policy logic is performed by the cloud-edge platform provider, or the user terminal. In the former case, the application running in the bottom two or all three layers of the architecture of Figure 3 is marshalled by the cloud-edge service provider and the end device is assumed to be under its control. An illustrative example for this is the case of industrial robots, where a robot control application is deployed in the infrastructure slice overarching all layers. The latter case is best represented by an autonomous mobile/cell phone user that buys cloud-edge resources in order to prolong the battery life of its device.

C. Processing Placement Requests

In the rows of Table III, we further divide the set of papers along the dimension of the temporal aspect of placement decisions, as introduced in Section III. In the "Offline" row those papers are depicted that assume a one-shot batch placement of all the tasks in hand. In the "Online" row, on the other hand, those solutions are gathered that solve the placement of each task one-by-one, preparing for a sequential arrival of requests for placement. The third category in this dimension, the row "Online with migration", groups those papers that apply task migrations while placing new tasks separately. This feature resembles to a re-optimization attempt, partially offline optimizing the already deployed tasks. Finally, making a complete step in this direction, "Hybrid" solutions fully

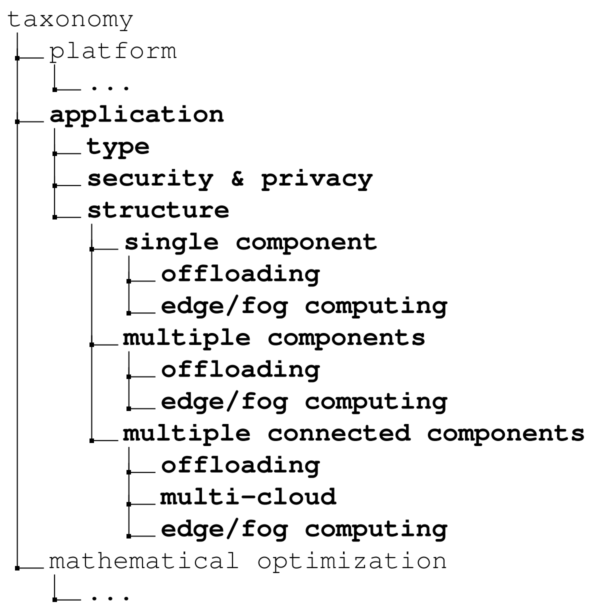


Fig. 5. The taxonomy angle analyzed in Section V.

re-optimize the placement status periodically, during which periods they operate as online methods.

One can find the same paper in multiple columns or multiple rows in Table III. The reason for that is the multi-facet results those papers contain. In a few cases, for example, researchers propose both an online and an offline method to address the placement challenge, without integrating them into a hybrid solution. The most usual reason for multi-column papers is that placement policy logic can be tied to both the cloud-edge provider and to the user device.

V. DEPLOYED APPLICATIONS

We present several collected papers in detail, and we underline the interesting specifics of the work published there. For tractability, we split the body of work into categories based on the second dimension group introduced in our taxonomy in Section III, i.e., service-related aspects, such as service composition the research papers assume, application type and security. As a reminder, we indicate the taxonomical aspects studied in this section by bold fonts in Figure 5.

We use the dimension **Service structure** as major categorization. The subsections are named respective to the possible compositions: most of the researchers consider single-component monolithic services, and there are research papers that address the placement of services that can be decomposed into sub-tasks, or must be replicated. We call these latter

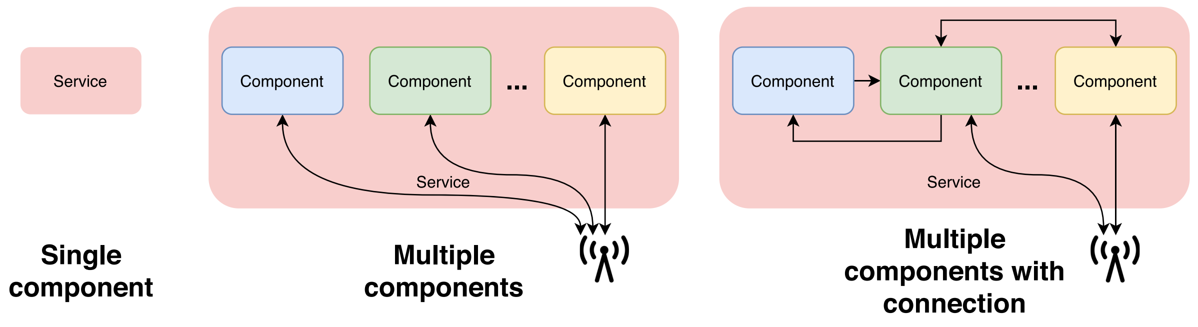


Fig. 6. Categories in terms of components and relations between them.

multi-component services. Recent works consider such multi-component services that also involve some notion of relations between the components: either network requirements are imposed, or data movement characteristics are modeled. The addressed service structures are illustrated in Figure 6 and all papers are listed in Table IV within their respective category. Listing the references within Table IV is meant to illustrate the balanced cut of the related papers via this choice of categorization, and helps the reader to query the respective category of any cited paper. Moreover, the distribution of the presented papers are presented graphically in Figure 7 as a section map with relevant statistics. In addition, a chronological overview of the surveyed papers is also given in Figure 9 highlighting the trends how the hot topics changed during the last six years. Within each subsection we further group the selected papers according to the **Application type** in focus, and pinpoint those that touch upon any **Security and privacy** aspects, following the other two dimensions in the second characteristic group defined in Section III. Papers addressing specific application types or security and privacy aspects are also highlighted and categorized in Tables V and VI, respectively.

A. Single Component

We split the papers targeting the placement of single component services in two classes. First, we group classic MEC offloading use cases together, second, we classify general edge or fog computing platforms' methods in one large group.

1) Offloading in MEC: In terms of **Platform components**, the papers that tackle offloading decisions always involve terminals, e.g., IoT devices, mobile user equipment. However, we can make a distinction among the collected papers, whether they focus only on edge servers in MEC, e.g., [81], [85], [94], [112], [113], [115], [117], [154], [157]–[160], [173], [183], or they also consider the possibility to offload computational tasks to a central cloud, e.g., [101]–[104], [148], [151], [152]. Furthermore, one can find papers in the related literature that apply moving devices, e.g., UAVs [84], or robots [173], as edge nodes in order to position them to optimal locations anytime the terminals change their location. In these latter the optimization usually involves finding those optimal locations, besides the offloading decisions. For example, in [84] the authors formulate a computational offloading problem among a swarm of UAVs acting as small flying cloudlets that receive compute-intensive tasks from IoT devices via the FDMA technique. UAVs, as MEC resources, aim to optimize their energy consumption while satisfying the requirements of the IoT tasks under execution. Similarly, related to the notion of cloudlet [79], [84], [173], a few papers aim at optimizing the location of edge nodes, hence performing the connected network design [79], [173]. The authors of [79] find the near-optimal places of the network elements and offloaded VMs for minimizing the overall network element installation costs and cloudlet access latency. Also in [173] a cloudlet placement problem is tackled in which mobile cloudlet robots are positioned such that they cover as many devices as possible to minimize the task processing time. The proposed model takes devices' mobility into account, so it determines the new locations for cloudlets and the shortest paths to get there. Not to confuse with the UAV-mounted edge servers, in [85] the authors propose cooperative computation offloading for UAVs into a heterogeneous edge platform, and they solve a latency constrained optimization problem by leveraging simulated annealing based particle swarm optimization approach.

In terms of subjects to optimization, the offloading decisions are often complemented with additional aspects, other than the aforementioned positioning of mobile edge nodes. Therefore, next to the research results about offloading decision optimization, e.g., [84], [94], [115], [148], [157], [160], [183], there are several works that propose to jointly optimize offloading and compute resource allocation in the edge infrastructure, e.g., [101], [102], [112], [113]. In these papers the authors strive to optimize the operation of the overall MEC system while end users and/or terminals aim at minimizing their own battery usage or the processing time that constitutes the service latency. In several papers network resources are also taken into consideration as subject of the optimization. In [144] the authors model the wireless channel, where the heterogeneous access points' capacities are derived from the dynamically allocated width of the spectrum and the signal strength. Similarly, radio transmit power [81], transmission data rate [85], momentary SINR [158], [159], wireless backhaul bandwidth partitioning [154], wireless link capacity [143] can be included in the optimization objective, or constraints when tackling the offloading decision making from the holistic system perspective.

The third dimension group of our taxonomy in Section III stands for the mathematical apparatus and formal modeling of

TABLE IV
CATEGORIES IN TERMS OF SERVICE STRUCTURE

Service structure	Single component (e.g. VM, container, task)	Multiple components (e.g. backups, replicas)	Multi components with connection (e.g. SFC, DAG)
Papers	[49], [54]–[57], [60], [63], [69], [74], [78], [79], [81], [82], [84]–[87], [90], [93], [94], [98], [101]–[104], [112], [113], [115], [117], [130], [143], [144], [148], [151], [152], [154], [157]–[160], [165], [166], [168], [173], [177], [181], [183], [184]	[51], [59], [64], [68], [70], [71], [75], [77], [80], [95], [105], [108], [110], [114], [127], [129], [131]–[133], [135]–[137], [140], [155], [161], [169], [170], [174], [176], [178], [180], [182]	[50], [52], [53], [58], [61], [62], [65]–[67], [72], [73], [76], [83], [88], [89], [91], [92], [96], [97], [99], [100], [106], [107], [109], [111], [116], [118]–[126], [128], [134], [138], [139], [142], [145]–[147], [149], [150], [153], [156], [162]–[164], [171], [172], [175], [179]

TABLE V
CATEGORIES IN TERMS OF APPLICATION TYPE

Application type	IoT	Industry 4.0	Big data analytics	Telco (or related) NFV
Papers	[55], [68], [73], [78], [84], [86], [88], [106], [111], [145], [153], [160], [163], [167], [175], [177]	[54], [76], [124], [146]	[59], [66], [67], [125]	[74], [95], [97], [124], [170], [180]

the collected body of research. The **Optimization objective(s)** therein is one of the most important characteristics that define a research effort. In the MEC offloading papers the goal is selected from a rather limited set of inherent choices: energy and processing time. One can find papers in which one of these two goals are set out, e.g., energy in [84], [85], [102], processing time in [94], [104], [115], [148], [152], [157], [173], and there are also related works in which the goals are targeted jointly, e.g., in [81], [103], [112], [113], [117], [183]. While the former goal aims at preserving the limited battery capacity of terminals, e.g., IoT sensors, mobile phones, the latter strives to reach a desired QoS level in terms of service response latency leveraging the compute capabilities of edge nodes in the proximity of the terminals. A few papers formulate general optimization goals as well, e.g., cost in [101], [148], revenue in [113], [158] where the authors propose to prioritize users with maximum utility to maximize the provider's revenue. In [148], the authors propose to optimize both the perceived latency and the service migration cost based on the computation demand and current position, assuming unavailable future system information and unknown system dynamics. They apply a contextual multi-armed bandit problem and a Thompson-sampling based online learning algorithm to explore the dynamic system environment. Also focusing on the experienced delay caused by invoking the migration of a VM, in [159] a VM placement with path selection and a migration scheduling using user mobility information are proposed. User mobility is modeled by the evolving SINR maps therein. Being an unusual optimization goal, in [160] the authors strive to solve the aggregating IoT gateway's capacity fragmentation issue caused by binary decisions of offloading the whole task or computing the whole task on the device. Instead of losing capacity due to this coarse-grained allocation, their optimization makes fine-grained offloading decisions at any stage of the process, also considering the energy required to send the data for processing on a nearby server. In [144] the optimization of resource allocation and computation offloading is extended with that of content caching for maximizing the profit of MEC system operators facing the end users, renting communication resources from mobile network operators. Inspired by a similar idea, in [94], [115] a cache-enabled fog computing network is proposed where fog nodes own storage capacities to proactively cache the popular tasks' results to minimize the computation time of the future task requests by returning the cached value. Further developing the potential of caching in the edge, the authors of [158] combine MEC and information-centric network architectures for optimizing the profit of a mobile virtual network operator by effective video transcoding, caching and multicast optimization.

When optimization affects multiple parts of the ecosystem, the researchers have to deal with even higher complexity than what is reached with a single objective optimization problem. The authors of [81] therefore split the main optimization task into the offloading decision and resource allocation sub-problems and sequentially solve them by leveraging one-to-many and one-to-one matching techniques. In [112] a centralized controller entity is suggested which decides about the tasks that should be offloaded and about the amount of resources to be assigned to them individually. The authors also decompose the original problem into a resource allocation problem with fixed task offloading decisions and a task offloading problem that optimizes the optimal-value function corresponding to the resource allocation problem. As the wide-area MEC architecture' backhaul link is assumed to be wireless in rural areas by the authors of [154], they propose an iterative algorithm to the offloading as a matching problem, the node capacity allocation as a linear programming problem, and finally, the bandwidth allocation as a univariate function minimization. As a great example for a distributed algorithmic solution, in [143] a distributed trading game is proposed in which the wireless devices are interacting with the network operator through a Stackelberg game for network and computing resources. Resource allocation and offloading decisions are made in the devices and the operator, while ensuring the preservation of the required wireless link capacity. The distributed offloading and resource allocation is proven to be efficient in the paper, which is used for the design of an approximation algorithm for the decision making.

In terms of **Application type**, IoT use cases, e.g., [84], [160], are over-represented in this MEC offloading collection of papers due to the low power nature of IoT

TABLE VI
PAPERS CONSIDERING SECURITY OR PRIVACY ASPECTS

	Security	Privacy
Papers	[86], [102]	[151], [183]

devices in general. Accordingly, some papers assume static users, e.g., [81], but those research results that target mobile terminals naturally take into account mobility as well, e.g., [157], [159], [173].

The remaining dimension of the second group in our taxonomy, **Security and privacy** appear in [102], [151], [183]. The authors of [183] points out that two potential privacy issues are induced by the wireless task offloading feature of MEC: location privacy and usage pattern privacy. Hence they propose a privacy-aware offloading mechanism to avoid the chance of being detected. Their solution decides where to run the tasks to achieve the best possible delay and energy consumption performance while maintaining the prespecified level of privacy. Somewhat different idea is investigated in [151]: the authors introduce a distributed auctioning model for the capacities, where low-latency applications are deployed with privacy constraints. The edge devices buy resources from their immediate neighbors, if their tasks are to be considered private. In [102] the authors consider different user-defined QoS requirements in terms of delay, compatibility and security. They apply Bender decomposition techniques to decompose the original offloading problem into a master problem and sub-problems that can be solved in parallel at fog nodes.

2) Resource Allocation in Edge and Fog Computing Systems: In this section we overview the research papers that focus on edge systems without considering the terminals' compute capabilities. In these papers the underlying infrastructure assumes a multi-cloud [49], a cloud-edge [54]–[57], [60], [63], [69], [74], [98], [165], [166], [177], [184], or a multi-edge scenario [78], [82], [86], [87], [90], [93], [130], [168], [181]. These choices belong to the **Platform components** dimension, and highly determine the optimization intention the researchers set out. When both a central cloud and edge computing are considered, in many cases the researchers assume that the workload is initially placed in the cloud, and the edge system needs to determine where to replicate and how to distribute the user load among them [55], [82]. Therefore they propose a framework to push the resource-intensive applications to the edge, to minimize average data traffic in the edge network among the base stations by replicating services from the cloud to a subset of the edge servers. The workload allocation over heterogeneous computing systems must take into account different resource availability [56], and when distributing the workload between fog and cloud, the goal might be to minimize the energy consumption of both of them, such that delay demands of the services are fulfilled [60]. If a given research work does not suppose the usage of a central cloud, i.e., tackles multi-edge scenarios, the challenge may stem from the joint optimization of device-to-base station association, task distribution, VM placement and resource allocation [57], [93]. The authors of [69] go further: they strive to minimize the user traffic load by addressing the joint service placement and user association, the joint allocation of computing and radio resources and the correlation of adjacent base stations' placement decisions.

While, in terms of **Optimization objective(s)**, most of the papers that fall into this category share the same target, i.e., latency of service completion [57], [60], [74], [78], [93], [166], [168], [181], [184], several research initiatives have been made to tackle the trade-off between power consumption and transmission delay as well, e.g., [56], [60]. Besides providing fast service completion to the users, researchers also try to cover as many users with the edge system as possible; the challenge is exacerbated by the fact that users move [49], [130], [181]. In order to do so, the authors of [130] formalize a time-variant and mobility-related optimization problem that accounts for varying position of the users leading to user re-allocations among different base stations to sustain user-perceived QoS. Their migration-enabled and mobility-aware approach, in which user migration is considered between adjacent base stations, aims to maximize the user coverage rate, to minimize the number of re-allocations and to yield refined dynamic allocations. A containers placement and migration strategy is proposed in [168] to maximize the number of satisfied users requests, with respect to delay QoS requirements and resources limits. Also recognizing the challenge caused by migrations, in [181] an energy-aware optimization scheme is proposed that minimizes the latency and the involved reallocation costs due to the limited edge server budget and user mobility. In general, cost minimization is a widely applied formalization in the research on edge computing platforms, and the cost can stand for different aspects, either resource consumption or service quality and its related revenue: in [93] the total deployment cost takes into account the wireless communication cost (link delay parameters are derived from the amount of data to be sent and the allocated wireless channel capacity) and the computation cost of function placement, in [63], [177] the cost is due to hiring edge servers that can ensure the required QoS for a maximized number of allocated users or requests, in [87], [98], [165] the cost is predicted for future service migration related to the mobile users within a look-ahead time window, and in [57] the service provisioning cost includes the number of VNF instances in the network (VNF sharing is enabled to lower the costs), and the transport bandwidth consumption as well.

The authors of [55] propose a placement and load distribution scheme for edge systems serving as many application requests as possible before their respective processing deadlines, minimizing the cost of task running, and maximizing the reliability of the provided applications. Reliability is emphasized in many other research papers [55], [74], [90], [166], [184] in this domain: edge nodes are, in general, prone to failures and it is considered to be slow and expensive to maintain them due their scattered geographic locations. Therefore authors of [166], [184] prepare for probable edge node failures by reserving backup placeholders for VNFs. The authors search the minimum amount of edge resources to be reserved in order to provide the necessary redundancy in the system for high reliability of services. In [184] they offer a Kubernetes-based solution for edge infrastructure, in which delay-sensitive applications can be deployed by the custom Kubernetes scheduler that makes its decisions with applications' delay constraints and edge reliability in mind. In [90] the authors maximize the total revenue collected by the service provider applying user ranking and admission control policy, and taking reliability requirements into consideration.

A recurring pattern specific to this category of the papers is the placement of gateways [49], [54], [78] within the edge platform. The trade-off between the insurance of QoS via the placement of VNFs of data anchor gateways closer to user devices and the avoidance of the relocation of mobility anchor gateways via the placement of their VNFs far enough from user devices is analyzed in [49]. The authors of [78] place fog nodes to the possible sites in a way that the overall latency between user gateways to fog nodes is minimized.

In terms of **Application type**, IoT [55], [78], [86], [177], Industry 4.0 [54], telco [74] and e-health [93] use cases revolve around the body of research in this domain. In the collected papers that fall into this category, only one tackles **Security and privacy**: in [86] the utilized bandwidth, required storage and the specified task security parameters are taken into consideration as constraints in the optimization.

B. Multiple Components

In this section we discuss those items of the related work that consider such edge applications that comprise multiple components. Those components can be, in general, deployed separately to different nodes, even in different layers, i.e., end device, edge, cloud. Similarly to Section V-A, we divide the papers targeting the placement of multi-component services in two classes. We group MEC offloading use cases in the first, and we classify general edge or fog computing platforms' methods in the second group.

1) Offloading: A relatively low number of published papers fall into this category. In each of the papers cited in this section, the resource allocation on edge servers is the main element of the study, and an optimal decision is sought for offloading some parts of the software from the end user device to the edge, so a centralized placement logic is designed for a cross-edge orchestration.

We divide the papers into two groups depending on how the authors model the application components to be placed.

In the first group, instead of directly placing VNFs, the workloads are modeled as individual tasks [71], [108], [114], [136], [155]. Within these models the service requests are single entities to be placed onto a heterogeneous edge computing network. The edge node's capacity can be segmented into slots, which is used to allocate the services [114]. Alternatively, the set of distributed task queues are matched to sparsely distributed cloudlets in a standard MCC architecture [71], or to execution nodes of a hybrid MCC environment consisting of a local cloudlet and other feasible mobile devices [136], or to UAV-mounted cloudlets [108], where the authors optimize not only task partitioning, task-to-UAV associations, and the allocated resources, but also the UAVs' positions. In all these related work the ultimate goal of the authors is to maximize the number of admitted tasks in the system. In [108], the optimization is performed subject to their required latency and reliability: the latency requirements are ensured by the locations of the UAVs, by the parallel computation of the split parts of the tasks, and by the radio and computational resources allocated for subtasks; reliability is provided by the number and size of the diverse subtasks and their associations to the UAV cloudlets. The authors in [136] consider the dynamic characteristics of both the incoming tasks and the computation providers, where the collaborating mobile devices with limited resources can join and leave the system arbitrary.

In the second group, the papers consider partitioned applications to be placed in the edge system [105], [110]: at the same time, multiple edge sites are able to work in parallel to get the result of the offloaded computational tasks. While the software is partitioned into unit blocks, the coordination to deploy and run those induces some edge site coordination cost. Moreover, mobile users may connect to various edge sites during their movement [110]. In terms of optimization goals, in [105] the authors strive to find the most energy efficient deployment of tasks, with the respective allocation of radio parameters, as the transmission costs of wireless end devices are mainly characterized by their energy consumption.

2) Resource Allocation in Edge and Fog Computing Systems: This section is devoted to the research works addressing services consisting of multiple components, such as VMs, containers or tasks, but there is no relation among the constituent elements taken into account. Following our **Platform components** dimension, the underlying infrastructure, where the service components are mapped to, can be multi-edge [80], [95], [129], [131]–[133], [135], [137], [140], [169], [180], [182], cloud-edge [59], [64], [68], [70], [75], [127], [161], [170], [174], [176], [178] or multi-cloud [51] but the core problem to be tackled, i.e., component placement, is similar.

Some research works deal with the optimization of the virtual infrastructure itself including, e.g., the placement of service-hosting VMs, or clustering of the fog resources. At a first glance, some parts of this problem are related to the task of network planning (which is out-of-scope of this survey), however, if the optimization happens on a shorter time scale assuming highly dynamic environments, that moves those research papers into the focus of this survey due to the similar characteristics of the underlying mathematical problem. For example, in [137], [180], the authors target the problem of dynamic placement of service-hosting nodes over a SDN-based, NFV-enabled MEC architecture in order to minimize operational costs. Authors of [137] focus on VNF replication capabilities, while [180] focuses on satisfying the service-level response time requirements. The latter paper presents an online adaptive greedy heuristic algorithm, which is also capable of managing the service elasticity overhead that comes from auto-scaling and load balancing with a proposed capacity violation detection mechanism. A similar infrastructure-related approach is followed in [174]. That study focuses on the medium-term planning of an edge cloud network in a MEC environment. The authors define a link-path formalization along with a heuristic approach for the placement of virtualization infrastructure resources and user assignments,

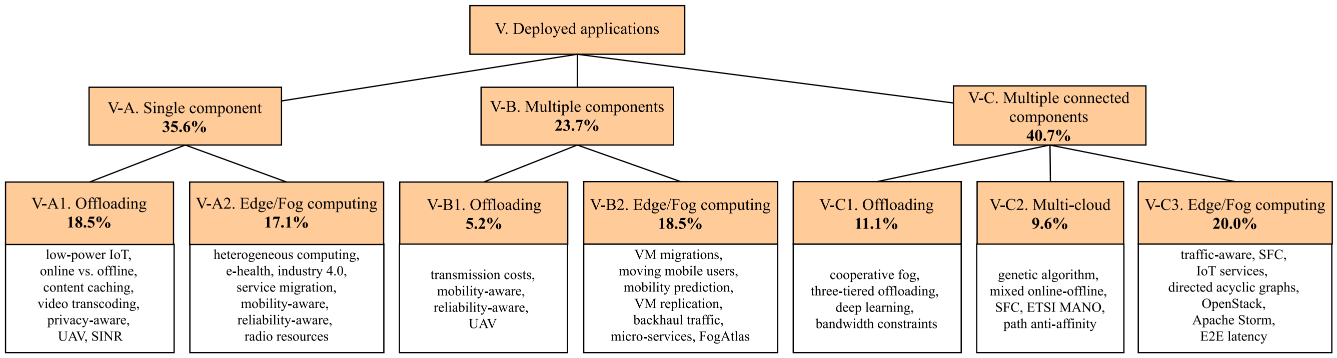

V. Deployed applications
V-A. Single component 35.6%
V-B. Multiple components 23.7%
V-C. Multiple connected components 40.7%
V-A1. Offloading 18.5%
low-power IoT, online vs. offline, content caching, video transcoding, privacy-aware, UAV, SINR
V-A2. Edge/Fog computing 17.1%
heterogeneous computing, e-health, industry 4.0, service migration, mobility-aware, reliability-aware, radio resources
V-B1. Offloading 5.2%
transmission costs, mobility-aware, reliability-aware, UAV
V-B2. Edge/Fog computing 18.5%
VM migrations, moving mobile users, mobility prediction, VM replication, backhaul traffic, micro-services, FogAtlas
V-C1. Offloading 11.1%
cooperative fog, three-tiered offloading, deep learning, bandwidth constraints
V-C2. Multi-cloud 9.6%
genetic algorithm, mixed online-offline, SFC, ETSI MANO, path anti-affinity
V-C3. Edge/Fog computing 20.0%
traffic-aware, SFC, IoT services, directed acyclic graphs, OpenStack, Apache Storm, E2E latency

Fig. 7. Distribution of the presented papers in the sections and some corresponding keywords.

i.e., determining where to install cloudlet facilities among sites, and assign access points, such as base stations, to them. In a similar vein, researchers in [75] suggest game-theoretic techniques for VM placement to ensure application's performance while they aim to jointly minimize infrastructure energy consumption and cost. Dedicated telco use cases are addressed with similar objectives, where the management cost of Telecom infrastructure vendors' network [170] and the 5G infrastructure [95] is to be optimized. In [95], researchers propose to formulate the edge device placement problem as a VNF placement task for reliable broadcasting in 5G RAN. The problem is formulated as a multi-objective optimization problem constraining bandwidth, service latency and processing capacity and minimizing the composite objective function for reliability, deployment cost and service response time. The particle swarm optimization and genetic algorithmic meta-heuristic approaches are used to solve the optimization problem.

The majority of the research works categorized into this section focus on service placement. However, some papers combine the optimization of the virtual infrastructure with the placement of service components. For example, the authors of [51] focus on the reliability aspects of placing VMs and their backup duplicates in a distributed cloud computing network. After the VM placement, as a second step, the service processing tasks are allocated to the reliably placed VMs using a maximum weight matching algorithm for bipartite graphs. The graph matching also takes into account the possible failure recovery strategies to assign tasks to VMs. In [135], the authors consider a coordinated planning of edge node location and VNF placement based on modeling the spatial and temporal mobile network usage over a geographic area. The proposed network slice planning framework has a modular structure, responsible for user mobility, service usage and edge cloud location. The predictive algorithm uses the spatio-temporal model for QoS-aware and load efficient VNF placement. A different approach is presented in [80]. The researchers propose a balanced clustering and joint resource allocation algorithm to achieve minimized response delay and energy consumption. They assume that the tasks are partitionable, and as such, they can be processed in parallel. The studied model contains multiple fog nodes communicating with each other via wireless channel. The proposed algorithm firstly clusters the fog nodes according to their wireless and computational resource and workloads to create balanced groups of servers. Secondly, it determines how to split the tasks into partitions and jointly allocates the wireless and computational resources in each cluster in parallel. The objective of their proposed method is to minimize the weighted latency and energy consumption cost of the worst offloaded task partition.

The surveyed research papers dealing with the placement of multi-component services take different aspects into consideration. A large part of the research efforts target applications realized by a configured number of replicas of given service components [51], [64], [70], [77], [132], [133], [137]. Replicas can increase the reliability, availability, dependability, and coverage of a service. However, in this context, the main goal of using replicas is to provide latency constrained service access for customers. For example, in [64] the authors assume that initially all the applications run in the cloud as VMs and the issue is how to deploy VM replica copies supporting multiple applications among numerous MEC servers in edge networks. Their objective is minimizing the average response time with various request demand and limited capacity of MEC servers in mobile edge networks by placing the VM replicas close to the users. Due to the edge environment, customers are generally connected via mobile networks, therefore, considering user mobility can also be essential. In several research papers [75], [131]–[133] [135], [140], [174], [180], [182], mobility patterns are incorporated in the models and the service deployment and/or resource allocation are optimized either proactively based on predictions or reactively based on measurements. For example, utilizing user mobility to meet the extreme low service latency requirements is studied in [132], where the trade-off of resource footprint versus application delay is formalized. Taking the user mobility pattern information and prediction as input, the most useful neighboring base stations are used as VM replication locations. The authors propose two algorithms, which reactively and proactively determine the replication locations, minimizing the service quality degradation due to on-demand VNF relocation. The authors in [131], [140] formulate both the offline and online versions of the corresponding optimization problem. The online algorithm determines the best matching between application components and edge/core servers using the Hungarian method, and then applies a local search procedure to consider the communication requirements along with the users' movements and the previous placement result to improve their solution. Another related aspect is migration which is considered by various models [133], [161], [169], [174]. The authors in [161] formalize the placement problem for microservices constructed from smaller containerized components. Their main objective is to minimize the total response time, considering eventual migration times of the microservice containers. The applied method is a Bayesian optimization-based reinforcement learning algorithm, which requires minimal monitoring and it is robust to noise. Combination of the aforementioned aspects is in the focus of [182], where the authors discuss the proactive deployment of service instance replicas among multiple edge nodes for managing the cost-efficient service migration based on the mobile users' movement trajectories.

The vast of the research works assume a dedicated central entity which is in charge of calculating the efficient placement of the service components. However, researchers in [127], [129] propose distributed algorithms. Authors of [129] provide a distributed resource assignment and orchestration algorithm which runs an agent on each involved computation node, and the application placement is determined as a result of a voting and election procedure. In [127], researchers propose a set of practical, uncoordinated strategies for service placement in edge-clouds and demonstrate that these techniques can perform well compared to optimal solutions in terms of response latency. The authors invoke the well-investigated problem area of resource storage allocation and the principles of different cache management techniques are applied advantageously to edge-computing environments.

In terms of **Application type**, a telco use case is addressed in [95], more specifically, reliable broadcasting services in 5G RAN are investigated. Another telco-related application is targeted in [180] and the proposed system supports mobile multimedia applications with low latency requirement. The authors of [170] also assume telco services and examine the problem of dynamic application placement considering variable user mobility patterns and a multi-tier cloud infrastructure incorporating cloud augmented Telecom nodes. A locally optimal algorithm is also proposed to reduce the operational cost by placing or moving the resident applications between different datacenters. The authors of [68] study the orchestration of IoT applications in cloud-fog computing environment. They assume a simple two-container cascade model which can cover a large set of IoT applications. The fog application is a cascade of a cloud module and a fog module. Research presented in [59] focuses on the service placement problem with data-intensive applications. To address this, the authors consider the data required for a latency-sensitive task as part of the service components to be placed.

The papers in this category, focusing on resource allocation in edge and fog computing systems, mainly consider revenue, utilization and/or delay as **Optimization objective(s)**. From the operators' perspective, optimizing for revenue, e.g., in [59], [68], [174], [180], or for utilization, e.g., in [127], [170], [176], [178], or for both revenue and utilization, e.g., in [129], [131], [140], is a straightforward decision. Some research works, such as [64], [132], [161], take the delay as the main subject of the optimization, which is also a reasonable choice as the quality of experience of the targeted edge applications are mainly determined by the perceived latency. Other researchers combine the delay with energy [80], with utilization [51], [70], [169] or with load balancing [135]. Alternatively, in [137] VNF replication is considered as the most characteristic operational cost. In [95], which is driven by a telco use case, reliability is also taken into account besides revenue and delay, which stems from the special requirements of such services. Not a usual objective is used in [133]. More exactly, the authors strive to minimize the overall backhaul traffic for wireless fog networks using strategies like VM migration and replication, focusing on the long term operation of the network. Another unique aspect of the optimization problem is emphasized in [176], [178]. Researchers argue that non-trivial amounts of data need to be stored in storage constrained edge servers to enable service execution, and that many emerging services exhibit asymmetric bandwidth requirements. To fill this gap, they study the joint service placement and request routing in MEC-enabled multi-cell networks with overlapping coverage regions of BSs and multi-dimensional (storage, computation and communication) constraints. The special needs are formalized in the optimization constraints.

C. Multiple Components With Connection

In recent years more and more researchers turned towards complex application models, in which the application is composed of multiple components, e.g., services, tasks, sub-tasks, and there is a relation to be modelled among them. This trend is also confirmed by the increasing number of references in the top right cells of Figure 9. The connection between components is generally tied to network resources, it can be, e.g., maximum allowed delay, average amount of network traffic, etc. Here we list those papers that fall into this category of application modeling. We observe the trend that by the time researches started to build such models, the hype of task offloading diminished, hence the lower number of papers in the next Section V-C1. On the other hand, the first wave of such models appeared in context of multi-cloud, either with a single-, or in a multi-operator setting. This is the reason why we dedicate the separate Section V-C2 even though we have not done the same for single-, and multi-component applications in Sections V-A and V-B. The chronological distribution of these papers is visualized in the top three rows in Figure 9.

1) Offloading: In this section, we overview the research papers addressing the offloading of distributed applications constructed by multiple connected (or more generally, related) components. Two types of underlying infrastructure are assumed by these papers. According to our **Platform components** dimension, the majority of the works focus on the more complex cloud-edge-terminal scenario, [99], [100], [106], [107], [142], [145], [147], [149], [150], [153], some others target the edge-terminal setup, [109], [111], [116], [156], while [139] presents a multi-edge scenario.

In the papers, two important aspects are considered: from the user's perspective, the experience is to be "maximized" by, e.g., minimizing the perceived latency or respecting a delay constraint; from the operator's point of view, the efficient utilization of resources (compute and network resources) is the main target. The research works presented in [100], [106], [111], [142], [145], [147], [153], [156] focus on the customers' aspects and optimize the user experience. For example, the authors of [142] propose a fine-grained computation offloading model, where parts of a computation task are considered for delegating to nearby computation nodes. An adaptation of the convex optimization, the alternating direction method of multipliers, is designed and detailed, which optimizes for user perceived QoS. The overall average service response delay of all users are minimized over the cooperative fog computing infrastructure model. Strategies of the cooperating nodes are also analyzed and the authors propose a solution based on their optimization. Other research works [99], [107], [109], [116], [139], [149], [150] jointly handle both the users' and operators' aspects and besides offloading decisions, the resource allocation is also considered. For example, in [149] researchers consider to integrate optimization of multi-tier offloading with resource allocation. In their first scenario, a two-tier offloading mechanism is assumed where the end devices' workload can only be processed either locally or in the cloud. The second scenario introduces the option to offload to the nearby edge computation nodes, making the offloading optimization three-tiered. In both scenarios the selection of the VNF deployment site is also chosen together with the offloading decisions. Multiple users' service requests are optimized simultaneously, where the wireless access capacity is divided among the tenants. In [150] the work of [149] is extended to support user requirements with strict delay constraints. The authors in [139] besides placement constraints, consider precedence constraints among the offloaded tasks for describing inter-task data communication in a multi-edge environment.

In terms of **Optimization objective(s)**, usually delay or utilization is targeted. In [100], [142], [156], the authors optimize solely for the delay, while in [149], [150] delay and energy are jointly controlled. The authors of [107], [109](online algorithm), [139], [153] optimize the utilization in their models while the delay is considered in the constraints. Of course, the two parameters can be combined, for example, in [99], [147], the delay and the utilization are jointly optimized. Moreover, in [99], [106], the revenue is also part of the optimization objective. Besides the delay, some works [99], [100], [106], [107], [111], [149], [150], [153] also take the bandwidth into consideration in the optimization problem as a constraint. In [109] (offline algorithm), [111], the authors' goal is to maximize the revenue.

In terms of **Application type**, performance-sensitive IoT applications [106], [111], [145], [153] and delay sensitive online games [107] are explicitly addressed by the surveyed papers, while most of the works assume general application models. An interesting use case is given in [147], the authors propose a cooperative artificial intelligence (AI) platform, where the tasks are deep neural networks, and the proposed solution calculates for all neural network layers the edge node where it should run, in order to provide the task's result as soon as possible.

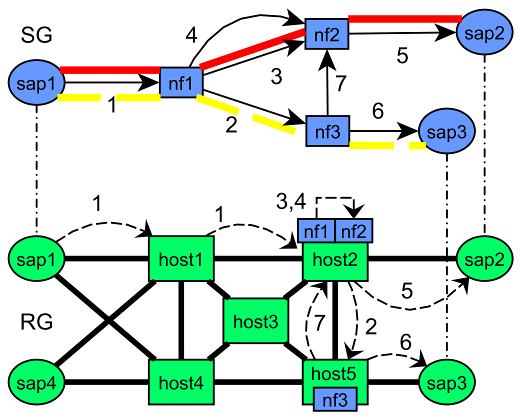


Fig. 8. Service Graph Embedding (SGE): a generalized variant of the Virtual Network Embedding (VNE) problem [123]. The goal is to find the optimal mapping of the service graph to the underlying resource graph.

2) Multi-Cloud: This section is devoted to the research papers falling into the multi-cloud category according to our **Platform components** dimension, while addressing complex services composed by connected/related elements. The typical problem to be solved is illustrated in Figure 8 (borrowed from [123]). The problem is referred to as Service Graph Embedding (SGE) or Service Function Chain (SFC) Embedding which is a generalized variant of the Virtual Network Embedding (VNE) problem.

Services or applications are modeled as graphs (service graph), where the network functions are the nodes of the graph and the connections/relations between the network functions are described by edges. A service graph (SG) is shown in the upper part of Figure 8 which consists of three network functions ($nf1$, $nf2$ and $nf3$) indicated by blue boxes. The attributes of these nodes describe the compute resource needs (e.g., cpu, memory, storage) of the corresponding computational units. The traffic flow between adjacent functions is represented by directed edges with unique IDs which can describe network related requirements, such as delay and bandwidth constraints. A request can optionally include Service Chains (SC), which define QoS requirements, such as maximal allowed latency or minimal bandwidth, on specific end-to-end paths of the service graphs. In Figure 8, SCs are denoted by red continuous lines and yellow dashed lines. The compute and network resources and related attributes are modeled by another graph, namely the resource graph (RG). An example RG consisting of five hosting nodes is depicted in the lower part of Figure 8. Users or other domains are connected by Service Access Points (SAPs). These SAPs indicate physical attachment points which are also referred to in the SG by special purpose nodes ($sap1$, $sap2$ and $sap3$ in the example). The overall goal is to find a mapping of the (virtual) nodes and edges of service graphs onto the shared physical substrate network, such that the cumulative resource allocations on any physical node or edge does obey capacity (and other) requirements. In Figure 8, an example mapping of the SG to the RG is shown, e.g., $nf1$ and $nf2$ are collocated on $host2$, while $nf3$ is mapped to $host5$, and the virtual links of the SG are assigned to indicated paths in the RG (the same link IDs are used as in the SG illustrating how given logical links are mapped to physical paths).

The vast of the papers [50], [52], [53], [72], [120]–[123] [171], [172] assume single-operator scenarios, i.e., the resources are owned and managed by a single provider, but some works [118], [119], [124] address the more complex and more challenging multi-operator setups. Of course, the later environments require business related questions as well to be investigated. Moreover, in order to enable multi-provider scenarios, **Security and privacy** aspects should also be considered. More exactly, the frameworks and data models should support hiding internal resource information between cooperating operators, for example, the internal network topology of a provider is not shared with the others and only abstract, high level resource information is exchanged. The surveyed research papers deal with general applications and service structures and they do not define limitations with respect to the **Application type**.

The majority of the proposed algorithms are online methods and only the authors of [50], [52], [53], [72] provide an offline mechanism. In addition, the researchers in [171], [172] propose an online-offline (hybrid) orchestration system, which receives and processes the incoming request in an online manner, and occasionally re-optimizes the total accumulated service deployment. The SGE problem is formalized as a mixed-integer linear program, solved using a greedy backtracking heuristic (online) and as an integer program (offline). For interleaving the simultaneous operation of the two algorithms, a framework is proposed where the strategic decision points are identified.

In terms of **Optimization objective(s)**, usually delay, revenue and utilization or some combination of them is targeted. In [120], [121], [172] only the revenue, in [118] only the delay, in [119] both delay and revenue are considered. In [171] delay and utilization, while in [123], [124] delay, utilization and revenue are taken into account in the optimization objective. Interestingly, in [50] the authors optimize for the bandwidth and in [72] the device power consumption is also included in the objectives. Authors in [52] study bandwidth optimization from multiple perspectives by minimizing aggregate bandwidth and minimizing the maximal link capacity utilization. Furthermore, the paper [53] studies cost minimization providing performance guarantees in relation to the optimal solution. In all papers, the constraints describe the capacity limits of the compute and network resources.

3) Resource Allocation in Edge and Fog Computing Systems: A similar embedding problem has to be addressed when the underlying infrastructure includes dedicated edge resources. Following our **Platform components** dimension, we distinguish the cloud-edge and the multi-edge scenarios depending on the placement options. The authors of [58], [61], [62], [65]–[67], [73], [76], [125], [126], [128], [162]–[164], [167], [175], [179] consider the cloud-edge scenario where the service components can be run either in the available edge domains or in the central cloud. Other research papers [83], [88], [89], [91], [92], [96], [134], [97], [138] investigate the multi-edge option where the central cloud cannot be used as a runtime environment.

Most of the papers assume single-operator scenarios where all resources, operated in central clouds or deployed to edge domains, are owned, managed and maintained by a single provider. This seems to be a realistic assumption and the first generation of edge and fog computing infrastructures follow (or will follow) this model. However, envisioned large scale, e.g., global scale, service deployments require compute and network resources from multiple providers and only a federation of operators can provision these services and applications. For example, in [58], [167], novel mechanisms for multi-provider scenarios are proposed and investigated.

There are several theoretical solutions proposed by the collected research papers, but some of them also provide prototypes or extensions to available cloud platforms. These extensions typically enable taking the network related aspects also into consideration, which is crucial in edge/fog computing systems. For example, the authors of [162]–[164], [167] (first architecture option) extend the widely used open source cloud management system, namely OpenStack, with network-awareness. More specifically, a novel online service placement solution is proposed that merges all the necessary functionalities for geographically distributed cloud-edge computing system under one common OpenStack domain. Their solution is capable of *i*) measuring the bandwidth and delay characteristics of the underlying physical network among compute nodes, *ii*) creating a topology model that contains both compute-, and network-related features, *iii*) mapping the incoming service requests, and re-mapping already deployed services to the underlying resources with their novel orchestration algorithm, iv) deploying and migrating services via OpenStack API calls. In an analogous manner, network-awareness has been added to different big data platforms. For example, the authors of [125] propose solving the resource allocation in the heterogeneous cloud-edge computing environment by extending the open source Apache Storm real-time computation system with delay-aware task placement. The paper details the correspondence of the request topology to a directed acyclic graph, built of Apache Storm platform components. It provides a good example of how theoretical results on service placement can be deployed on top of an adopted architecture. Similar extensions to HDFS and Spark are proposed in [66] and [67], respectively.

In terms of **Application type**, the previously highlighted papers describing extensions to big data platforms [66], [67], [125] can enable novel big data analytics applications. Other papers [73], [76], [88], [146], [163], [167], [175] focus on IoT or Industrial IoT applications. For example, the authors of [88] aim to jointly place the data processing IoT application and its subsidiary VNFs over a set of cloudlets and gateway nodes in a cost efficient way. Authors of [146] consider a cloud robotics warehousing use case, falling in the Industry 4.0 application type. Coverage and battery consumption constraints are modeled and taken as input to the optimization, so the usage of excess mobile computation capacity is enabled. However, most research works target general applications and service structures. The work in [65] focuses on modeling the characteristics of application-specific network slices for 5G use cases, and applying it to VNF placement optimization. They model VNF

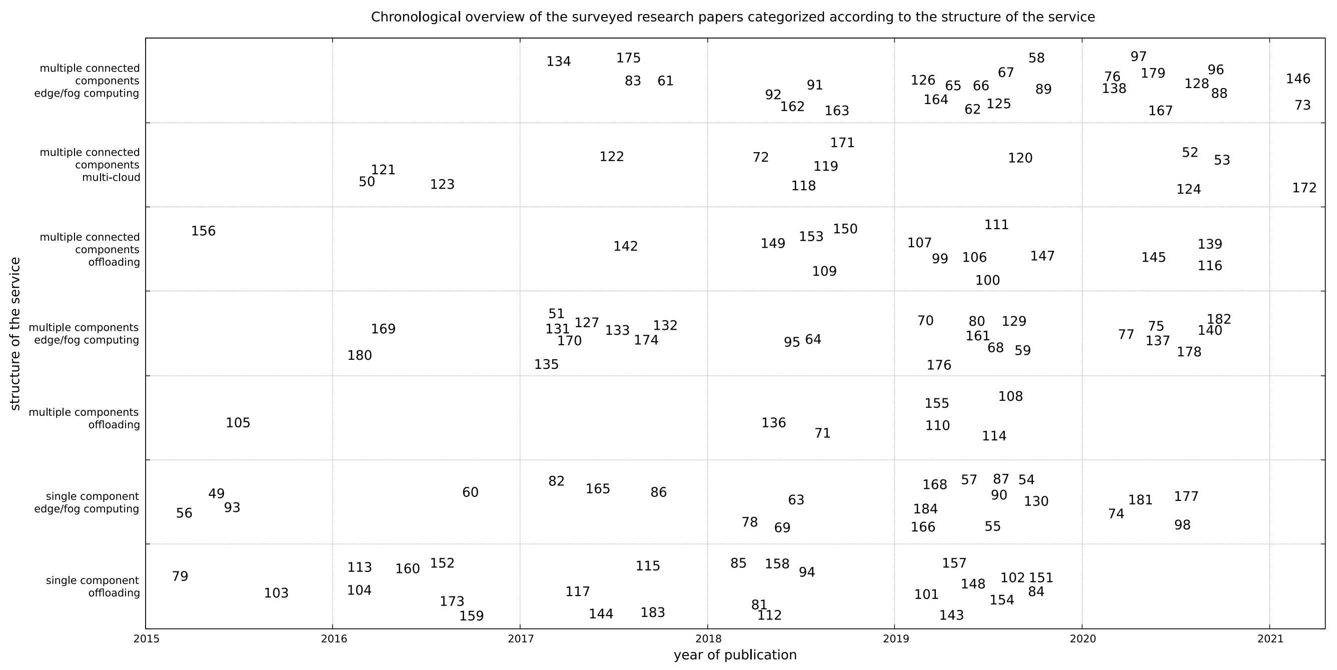


Fig. 9. Chronological overview of the surveyed research papers categorized according to the structure of the service (followed in Section V). The references in corresponding cells are randomly distributed and the exact publication dates within the given year are not reflected by the placement of the identifiers.

interference as the negative effect of collocated VNFs to their individually provided services. The performance degradation is studied in terms of increasing number of collocated VNFs, and a heuristic algorithm is proposed, which exploits this knowledge for VNF placement optimization. Evaluation scenarios of video streaming and autonomous driving are considered over a virtualized, software-controlled 5G network. Researchers in [97] investigate another telco use case and devise an energy-saving and resource-efficient VNF placement algorithm for network operators' architectures based on the ETSI MANO framework. Their proposed solution strives for saving energy and satisfying the placement demands by adjusting the scale of the substrate network and powering on or off the servers according to real-time load.

The **Optimization objective(s)** in the surveyed papers addressing cloud-edge and multi-edge platforms are slightly different from the ones applied for multi-cloud systems. As edge resources are scarce and expensive, therefore the main objective terms are utilization and revenue, while delay is considered mainly as a constraint. Interestingly, in [62], [73], [76], [91], [125], [138], the target of the optimization is solely the delay. For example, the authors of [91] focus on delay-critical services, while in [125] big data applications are addressed, where delay is the most important factor explaining the special role of that in the optimization problem. (The authors of [67], also dealing with big data use cases, combine delay, utilization and bandwidth in their optimization objective.) Researchers in [73] examine the optimal joint placement of data processing operators and pub/sub brokers in an IoT use case, where the sum of the end-to-end delays perceived by the subscribers is to be minimized. The authors of the paper [62] targeting the delay as the optimization objective follow a different way than other works. They propose using constraint programming to address the service placement problem in fog computing infrastructures, instead of integer linear programming and heuristic solutions. They argue that constraint programming is more generic and easy-to-upgrade model for an adaptive system, where constraints and objectives can change dynamically. They assume that the services can be written as a service graph and between the service components exist network constraints, like the minimum bandwidth or maximum latency that the application tolerates. The authors show that their proposed implementation provides a good trade-off between resolution times and solutions quality.

VI. Algorithmic Models and Solutions

In this section, we present the mathematical achievements of the collected papers, somewhat abstracted from their actual use cases, in terms of architecture and service choices. We first list the formalization tool sets of problem statements the researchers opted for. Second, partly related to the problem formulation, we set out the types of methods researchers applied to solve those problems. Third, again related both to the problem formulation and the targeted use case, we specify the exact optimization goals, and fourth, we briefly present the considered constraints, as well. Finally, for those collected papers in which the authors disclose algorithmic complexity along the presented methods, we show their high level summary for comparison. As a reminder, we indicate the taxonomical aspects studied in this section by bold fonts in Figure 10.

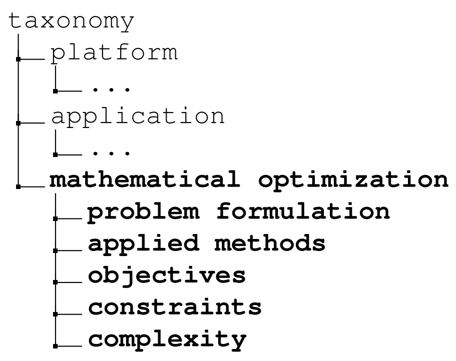


Fig. 10. The taxonomy angle analyzed in Section VI.

A. Problem Formulation

The problem formulation leads to abstract models that turn the applications or services, or even functions and tasks of those, into entities that need placement. In some papers, these entities are VMs, containers, or for example, pods in the realm of Kubernetes [6]. Nevertheless, the models are the abstract translations of the actual entities into, most often, combinatorial problems, in which the placement decision is to be made on possibly multiple layers of our architecture of Figure 3, and from a number of nodes/hosts within those layers. The placement options to select from range from devices, across edge nodes to cloud data centers.

Usually the combinatorial placement problem, i.e., matching entities to places, is coupled with and/or triggers decisions about offloading, resource provisioning, and scheduling. Therefore, in general, the addressed problems are multi-faceted, and therefore they inspire researchers to apply various tool sets for mathematical formalization. In Table VII and VIII, we list the formalization approaches we have encountered in the collected papers, and we assign each paper to its respective specific model type. The left-most two columns in both Tables VII and VIII depict the large categories and the specific models of problem formalization, respectively.

As it is reflected by the number of papers in those categories linear and nonlinear programming are the most widely used formalizing frameworks. Both categories are considered special cases of mathematical programming (also known as mathematical optimization), particularly linear programming, in which the goal is the optimization of a linear objective function, subject to linear equality and linear inequality constraints. In nonlinear programming some of the constraints or the objective function itself are nonlinear. By looking further into the specific models in the second left-most column of Table VII, one can learn that combinatorial optimization is the most prominent formalization researchers choose to define the placement problem. This choice is straightforward given the nature of the problem in focus, i.e., how discrete structures, the tasks and the infrastructure hosts, can be arranged together. All the integer programming models listed in the table, i.e., ILP, MILP, INLP, MINLP, MIQCP, are some sort of combinatorial problems: mathematical optimizations in which some or all of the variables are restricted to be integers. An illustrative MILP formalization example is highlighted at the end of this section.

In the papers that we classify into the General mathematical programming category in Table VII, the authors do not state whether their formalization falls into linear or nonlinear programming, instead, the formalization is described in a general form of mathematical optimization. Contrary to those, in Table VIII we group those papers that explicitly name the chosen formalization category. Among the selected mathematical frameworks, we find constraint programming, graph theory, game theory, stochastic optimization, optimal control and matching theory. The paper using a formalization of any of these categories constitute the minority of the body of related work. Many of those are combinatorial in nature, e.g., graph and matching theory, constraint programming. For example, in [62], the authors propose constraint programming instead of ILP and related heuristics and they argue that constraint programming yields a more generic and easy-to-upgrade model for an adaptive system, where constraints and objectives can change dynamically. Furthermore, there are several papers that involve stochastic components in their model, e.g., a randomly defined process of job arrivals, mobile users' movements, etc., and hence they turn to some sort of stochastic optimization approach. The variety of the specific models is rich, among those Optimal stopping, Markov decision process and Multi-stage stochastic programming are all concerned about making the optimal decision at a time or state or stage, respectively, of the process with the goal of optimizing the expected value of the reward, i.e., target function, in the future. For example, the authors of [157] apply optimal stopping theory to minimize the execution delay in a sequential decision manner. The other two models, i.e., Multi-armed bandit and Stochastic knapsack problem, tackle the stochastic nature of imperfect information about the model parameters. In the former, the reward, i.e., the target function, is only partially known at the time of allocation, in the latter the rewards are deterministic but the sizes are random, therefore the uncertainty is modelled on the constraints' side. Also tackling the temporal aspect of placement decisions, authors of a few papers apply optimal control theory for finding a control law for the dynamical system over a period of time such that an objective function is optimized. Specific models include Lyapunov optimization and Job shop scheduling, the latter being one of the best known combinatorial optimization problems. As a few examples for the former, Lyapunov optimization techniques are proposed to obtain asymptotic optimum with battery capacity of mobile devices stabilizing around a positive constant in [110], to minimize the upper bound of a queuing system in [155], for location prediction combined with the deep learning method of long short-term memory (LSTM) in [87], [98].

Besides these traditional problem formulating frameworks, several papers include game theory in their modeling section, a rarely seen model in placement problems. In those papers, placement decisions are made as a result of a strategic interaction among rational decision-makers, e.g., task owners and computing host providers. When the interaction involves sequential decision making from the participating players, then a Stackelberg game is applied for the model. For illustrative examples, we list a few of those papers. In [84] a potential game is formulated, and solved by an iterative algorithm,

TABLE VII
CATEGORIES IN TERMS OF MATHEMATICAL PROBLEM FORMULATION AND APPLIED TECHNIQUES (PART 1: MATHEMATICAL PROGRAMMING)

Formalizations		**Applied methods in papers and specific techniques**		
Category	**Specific model**	**Method**	**Papers**	**Specific technique** (if given/relevant)
Linear programming	LP	Solver	[104]	
		Convex optimization	[80]	Interior-point method
			[142]	Alternating direction method of multipliers
		Assignment method	[126], [128]	Hungarian method
	ILP	Solver	[58], [63], [120], [175]	
		Linear programming	[88], [176], [178]	LP with fractal solution rounding
		Graph theory	[118]	Shortest path
			[51], [136]	Bipartite graph matching
		Game theory	[84]	Best response search
			[97]	Matching game
		Search algorithms	[73], [106]	Tabu search
		Evolutionary algorithms	[182]	Particle swarm optimization
		Other heuristic solution	[54], [58], [74], [83], [90], [103], [105], [107], [111], [121], [163], [177], [180]	
		n/a	[49]	
	MILP	Solver	[57]	
		Linear programming	[59]	LP with fractal solution rounding
			[52], [53]	Column generation
		Convex optimization	[117]	Semidefinite relaxation
		Stochastic control	[127]	Cache replacement policies
		Assignment method	[131], [140]	Hungarian method
		Search algorithms	[79]	Branch and bound
		Other heuristic solution	[91], [146], [169], [171], [172], [174]	
Nonlinear programming	INLP	Other heuristic solution	[100]	
	QCQP	Convex optimization	[149], [150]	Semidefinite relaxation
	MINLP	Solver	[86]	
		Assignment method	[56], [60]	Hungarian method
		Convex optimization	[56], [112]	Convex and quasi-convex optimization
			[101]	Geometric programming
			[60]	Interior-point method
		Linear programming	[60], [102], [116]	Benders decomposition
			[86]	LP with fractal solution rounding
		Stochastic control	[108]	Sine cosine algorithm
		Other heuristic solution	[68], [81], [93], [112], [139], [154]	
		n/a	[113]	
	MIQCP	Other heuristic solution	[166]	
Mathematical programming (general)		Solver	[151]	
		Graph theory	[165], [173]	Shortest path
			[156]	Constrained graph partitioning
		Machine learning	[78], [173]	k-means clustering
		Evolutionary algorithms	[55], [70], [95], [99], [119], [137], [138]	Genetic algorithms
			[85], [95], [179]	Particle swarm optimization
			[71]	Hybrid queue Ant-Bee colony optimization
		Search algorithms	[109]	Backward induction
			[170]	Iterative local search
		Stochastic control	[161]	Bayesian optimization
		Linear programming	[114]	LP with fractal solution rounding
		Game theory	[89]	Iterative combinatorial auction
		Matching theory	[69]	Deferred acceptance
			[76]	
		Other heuristic solution	[64], [72], [77], [82], [96], [122], [129], [130], [132], [133], [135], [144], [159], [160]	

resulting in a pure-strategy Nash equilibrium, at which no participating player can reduce its cost by unilaterally changing its strategy. The authors of [94], [115] propose a matching game, in which a cost function of computing delay is minimized under latency and reliability constraints. The study in [152] comprises a two-tier trading game, in which data service subscribers pay for the resources of massive data center operators, while disclosing their QoS requirements, in turn, the latter purchase resources from the fog nodes if needed, which can provide data services with low delays to the former. For the

TABLE VIII
CATEGORIES IN TERMS OF MATHEMATICAL PROBLEM FORMULATION AND APPLIED TECHNIQUES (PART 2)

Formalizations		**Applied methods in papers and specific techniques**		
Category	**Specific model** (if relevant)	**Method**	**Papers**	**Specific technique** (if given/relevant)
Constraint programming		Solver	[62]	
		Other heuristic solution	[158]	
Graph theory	Virtual network embedding	Linear algebra	[50]	Eigendecomposition
		Graph theory	[145]	Graph partitioning
		Other heuristic solution	[65]–[67], [123], [124], [162], [167]	
Game theory	Game theory	Game theory	[84]	Best response search
			[75]	Auction theory
		n/a	[49]	
	Stackelberg games	Other heuristic solution	[143]	
		n/a	[152]	
Stochastic optimization	Multi-armed bandit	Search algorithms	[148](on)	Thompson sampling
	Optimal stopping theory	n/a	[157]	
	Markov decision process	Machine learning	[168]	Deep reinforcement learning
		n/a	[183]	
	Stochastic knapsack problem	Other heuristic solution	[134]	
	Multi-stage stochastic programming	Approximation	[181]	Sample average approximation
Optimal control	Lyapunov optimization	Machine learning	[110]	Sampling and classification
			[87], [98]	Recurrent neural network
		Graph theory	[87]	Shortest path
		n/a	[155]	
	Job shop scheduling	Evolutionary algorithm	[147]	Genetic algorithms
Matching theory		Matching theory	[115]	Deferred acceptance
		Other heuristic solution	[81]	

TABLE IX
HARMONIZED NOTATIONS USED IN [124] AND [171]

Notation	Description
R= (V_R, E_R), S= (V_S, E_S)	Resource and service graph
$\mathcal{P}(S)$	Power set of set S
$\mathcal{R}$, $\mathcal{T}$	Set of node resources and VNF types of R and S
$c_R : V_R \times \mathcal{R} \mapsto \mathbb{R}^+$	Resource capacity of a node in R
$c_S : V_S \times \mathcal{R} \mapsto \mathbb{R}^+$	Resource requirement of a VNF
$\tau_R : V_R \mapsto \mathcal{P}(\mathcal{T})$	Supported VNF types of a node
$\tau_S : V_S \mapsto \mathcal{T}$	Functional type of a VNF
$d_R, b_R : E_R \mapsto \mathbb{R}^+$	Delay and bandwidth characteristics of R links
$d_S, b_S : E_S \mapsto \mathbb{R}^+$	Delay and bandwidth requirement of S links
$\mathcal{C}^S \subset \mathcal{P}(E_S) \times \mathbb{R}^+$	Path delay requirements of S
$e_j, j \in E_S \times V_S = \Psi_S$	Leg, the unit orchestration step
$\mu : \Psi_S \cup V_S \mapsto V_R$	Hosting node of service node
$\lambda : \Psi_S \cup E_S \mapsto \mathcal{P}(E_R)$	Hosting path of service link
$\mathcal{N} : \mathcal{P}(E) \mapsto \mathcal{P}(V)$	Nodes of a path (in resource or service graph)

TABLE X
MATHEMATICAL NOTATIONS USED IN FORMULATION 1 [171]

$x^i_u \in \{0, 1\}$	Variable indicating mapping of $i \in V_S$ on $u \in V_R$
$y^{i,j}_{u,v} \in \{0, 1\}$	Variable indicating mapping of $(i, j) \in E_S$ on $(u, v) \in E_R$
$a_{u,r} \in \mathbb{R}_{\geq 0}$	Variable equaling total node allocations for $u \in V_R$ and $r \in \mathcal{R}$
$a_{u,v} \in \mathbb{R}_{\geq 0}$	Variable equaling total bandwidth allocations on $(u, v) \in E_R$
$U^{\min}_{V_R} \in [0, 1]$	Variable indicating the minimum node utilization
α, β, γ	Coefficients for normalizing and weighting objective function
$p_{i,u} \in \mathbb{R}_{\geq 0}$	Price for placing service node $i \in V_S$ on resource node $u \in V_R$
$p_{u,v} \in \mathbb{R}_{\geq 0}$	Price for using bandwidth along resource edge $(u, v) \in E_R$

resulting Stackelberg games a distributed price setting and purchasing strategy is designed for each actor, which is proven to reach Nash equilibrium, realizing an efficient resource allocation. Reference [49] two solutions favor one objective over the other, whereas the third one aims at finding a fair trade-off between the two objectives by the use of bargaining Nash theory.

1) An Illustrative Mathematical Programming Example: As an illustration, the MILP formalization proposed in [171] is summarized here briefly, which is the offline part of a hybrid optimization method. The general notations and the MILP variables are introduced by Tables IX and X, respectively, while the MILP is described by Formulation 1.

Resource and Service Graphs are denoted by $R = (V_R, E_R)$ and $S = (V_S, E_S)$ which describe the substrate infrastructure and the incoming request, respectively. It is worth noting that the model used in [171] assumes that service nodes (or functions) have types which can be hosted by only resource nodes supporting the given types. Available types are given by set $\mathcal{T}$ while the set of resources (available or requested cpu, memory or storage) is indicated by $\mathcal{R}$. For the power set of set S, we use the regular $\mathcal{P}(S)$ notation. c_R and c_S denote the resource capacity of a node in R and the resource requirement of a VNF in S, respectively. The functional type of a VNF is given by τ_S, while the supported types of a hosting node is indicated by τ_R. The network characteristics (link delay and bandwidth) are described by edge attributes d_R and b_R in the substrate graph. The counterparts are the delay (d_S) and bandwidth (b_S) requirements in the request graph. Delay requirements can also be defined for arbitrary paths in S, which are collected by $\mathcal{C}^S$.

Each node of a path is referred to as $\mathcal{N}$. The mapping result is defined by μ and λ describing the hosting nodes for VNFs and the hosting paths for logical links. e_j, j are explained in Section VI-B.

As we have illustrated in Section V-C2, the general task of SGE/VNE is to find a mapping of the service graph $S = (V_S, E_S)$ to the resource graph $R = (V_R, E_R)$. In the MILP formulation of [171], the node and link mappings are represented by binary variables $x_u^i \in 0, 1$ and $y_{u,v}^{i,j} \in 0, 1$, i.e., $x_u^i = 1$ indicates that service node *i* is mapped on resource node *u* and $y_{u,v}^{i,j} = 1$ indicates that the resource link $(u, v) \in E_R$ is used to establish the service link $(i, j) \in E_S$. The objective function is defined by Equation (1) in Formulation 1, where the first two summands describe the costs for resource allocations on edges and nodes respectively, while the third one is used for load balancing. A specific scaling factor is used for each summands ($\alpha, \beta, \gamma \geq 0$) in order to (*i*) normalize and (*ii*) weight the different components of the objective. The first summand expresses costs for using bandwidth by employing prices $p(u, v) \geq 0$ which can be set by the provider according to the importance of the respective links. The second summand expresses costs for mapping a service node $i \in V_S$ onto a resource node $u \in V_R$ using prices $p(i, u) \geq 0$. For the last term, controlling the load balancing, an additional variable $U_{V_R}^{\min} \geq 0$ is used, which denotes the minimum (node) resource load among all resource nodes and resource types. Constraint (7) upper bounds this variable by the allocations $a_{u,r}$ of resource $r \in \mathcal{R}$ on node $u \in V_R$ divided by the respective capacity. Hence, $U_{V_R}^{\min}$ must be less than the (relative) load with respect to any node and resource. Hence, by minimizing $(1 - U_{V_R}^{\min})$, the minimum load shall be increased, leading to distributing load more evenly.

Typical constraints are formalized in Equations (2)-(10). Constraints (2) and (3) enforce that each service graph node is mapped onto a suitable resource graph node while forbidding mappings to nodes that do not support the respective function type. Constraint (4) induces a unit-flow for each service link $(i, j) \in E_S$ using the flow variables $y_{u,v}^{i,j} \in \{0, 1\}$ for all resource edges $(u, v) \in E_R$. The left-hand side of the constraint states flow preservation, while the right-hand side enforces the sending of a unit flow from the node onto which the tail node *i* is mapped while the node onto which the head *j* is mapped must receive a unit of flow. Note that, when both *i* and *j* are mapped to the same node $u \in V_R$, then no network path needs to be established in the resource graph. Constraints (5) and (6) compute the allocations induced by the node and link mapping, respectively, and Constraints (8) and (9) enforce that these allocations are upper bounded by the capacities of the respective resource graph elements. Assigned nodes consume computation resources (e.g., CPU, memory, storage), while embedded links result in assigned bandwidth on respective substrate links. To enforce latency constraints for the set of chains $\mathcal{C}^S$, Constraint (10) is used. For each tuple $(p, D_p) \in \mathcal{C}^S$, the sum of delays of all resource edges used by any of the service links is computed (left-hand side) and is upper bounded by the maximum allowed latency D_p (right-hand side).

Formulation 1 Service Embedding MILP From [171]

$$\min \quad \begin{pmatrix} \alpha \cdot \sum_{(u,v) \in E_R} a_{u,v} \cdot p(u,v) \ + \\ \beta \cdot \sum_{i \in V_S, u \in V_R} x_u^i \cdot p(i,u) \ + \\ \gamma \cdot (1 - U_{V_R}^{\min}) \end{pmatrix} \tag{1}$$

$$\sum_{u \in V_R, \tau_S(i) \in \tau_R(u)} x_u^i = 1 \qquad \forall i \in V_S \tag{2}$$

$$\sum_{u \in V_R, \tau_S(i) \notin \tau_R(u)} x_u^i = 0 \qquad \forall i \in V_S \tag{3}$$

$$\sum_{(u,v) \in \delta_u^+} y_{u,v}^{i,j} - \sum_{(v,u) \in \delta_u^-} y_{u,v}^{i,j} = x_u^i - x_u^j \quad \forall (i,j) \in E_S, u \in V_R \tag{4}$$

$$\sum_{i \in V_S} x_u^i \cdot c_S(i,r) = a_{u,r} \qquad \forall u \in V_R, r \in \mathcal{R} \tag{5}$$

$$\sum_{(i,j) \in E_S} y_{u,v}^{i,j} \cdot b_S(i,j) = a_{u,v} \qquad \forall (u,v) \in E_R \tag{6}$$

$$a_{u,r}/c_R(u,r) \geq U_{V_R}^{\min} \qquad \forall u \in V_R, r \in \mathcal{R} \tag{7}$$

$$a_{u,r} \leq c_R(u,r) \quad \forall u \in V_R, r \in \mathcal{R} \tag{8}$$

$$a_{u,v} \leq b_R(u,v) \quad \forall (u,v) \in E_R \tag{9}$$

$$\sum_{(i,j) \in p, (u,v) \in E_R} y_{u,v}^{i,j} \cdot d_R(u,v) \leq D_p \qquad \forall (p, D_p) \in \mathcal{C}^S \tag{10}$$

2) An Illustrative VNE Example: Besides mathematical programming, invoking graph theory is reasonable especially for services consisting of multiple components with connections (and, e.g., latency requirements) among them. When the underlying infrastructure also exhibits specific delay and bandwidth characteristics, then the mathematical problem is to find the mapping / embedding between two graphs, namely, the service graph and the resource graph. Generally, the formal models considering all practical aspects and constraints yield quite complex optimization problems, thus, a great variety of heuristic algorithms have been proposed by the research community to approximate the optimal solutions. Here, we highlight the main ideas from [124]. The paper proposes an efficient online embedding algorithm which operates in a multi-layer orchestration hierarchy and supports several types of constraints, such as end-to-end QoS characteristics, cost limits and reliability requirements. The mathematical problem is formulated as a variant of VNE, where the underlying topology, resources and capabilities are modeled by a resource graph $R = (V_R, E_R)$, while the service deployment requests, encompassing multiple constituent components with connections and constraints among them, are modeled by service graphs $S = (V_S, E_S)$. Formulation 2 presents the online VNE problem from [124], where only a single service graph is considered and some requirements (e.g., node and link (anti-)affinity) are omitted. The notations are summarized in Table IX.

The mapping structures μ, λ describe a mapping solution, where each service graph node V_S is mapped to a node in the resource graph V_R. The link mappings must be valid, i.e., their hosting paths must start and end at the hosts of their ends as described by Equation (11). The node mapping must respect the functional requirements of the VNFs $i \in V_S$ as required by Equation (12). The constraints are quite similar to

Formulation 2 Optimization Problem From [124]

$$\forall (i,j) \in E_S : \mu(i) = \lambda(i,j).first \quad \text{and} \quad \mu(j) = \lambda(i,j).last \tag{11}$$

$$\forall i \in V_S : \tau_S(i) \in \tau_R(\mu(i)) \tag{12}$$

$$\forall u \in V_R, \forall r \in \mathcal{R}: \sum_{\{i|\mu(i)=u, i \in V_S\}} c_S(i,r) \leq c_R(u,r) \tag{13}$$

$$\forall (i,j) \in E_S : \sum_{(u,v) \in \lambda(i,j)} d_R(u,v) \leq d_S(i,j) \tag{14}$$

$$\forall (u,v) \in E_R, \quad M(u,v) := \{(i,j) | (u,v) \in \lambda(i,j) \text{ and } (i,j) \in V_S\} : \sum_{(i,j) \in M(u,v)} b_S(i,j) \leq b_R(u,v) \tag{15}$$

$$\forall (p, D_p) \in \mathcal{C}^S : \sum_{(i,j) \in p} \sum_{(u,v) \in \lambda(i,j)} d_R(u,v) \leq D_p \tag{16}$$

$$min_{\mu,\lambda} \sum_{e_j, j \in \Psi_S} \textsc{CalcObjectiveValue}\Big(\lambda(e_j, j), \mu(e_j, j), e_j, j\Big) \tag{17}$$

the ones used in Formulation 1. More exactly, Constraint (13) ensures that node capacity requirements mapped to a resource node $u \in V_R$ do not exceed the total capacity of the node for each resource type $\mathcal{R}$. Link-wise delay requirements must be respected by the link mapping function λ for each VNF connection $(i,j) \in E_S$ as stated by Constraint (14). Constraint (15) defines the set of all VNF connections using a substrate network connection as $M(u,v)$. This set is used to summarize all the bandwidth capacity requirements $b_S(i,j)$, which must be upper bounded by the resource link's bandwidth capacity $b_R(u,v)$. Besides the link-wise delays, path delay requirements $\mathcal{C}^S$ are also added which define maximal allowed latency on multiple consecutive VNF connections of E_S (see Constraint 16). The general objective function, which minimizes the sum of the objective value of each VNF $j \in V_S$ and the adjacent service graph connection $e_j \in E_S$, is formulated by Equation (17) and will be discussed later. Most of the practically interesting variants of the VNE problem are known to be $\mathcal{NP}$-hard and strongly inapproximable [185]. This variant introduces more constraints, such as the path delay requirement, so the same observations about complexity apply to that VNE formulation, therefore, heuristic solutions are needed.

B. Applied Methods

A myriad of methods are proposed to solve the formulated problems in the related papers. In the middle column of Tables VII and VIII, we specify the large group of the applied method for each paper, and if available, we denote the specific technique in the rightmost column of both tables. Generally speaking, most of the papers either use solvers or authors propose their own heuristics to solve the modeled problem. Similarly frequently seen, various well-known assignment, e.g., Hungarian, and search, e.g., Branch and bound, methods are applied either on their own, or as a basis for custom heuristics. The third most often used group of methodologies includes linear, e.g., Benders decomposition, and convex, e.g., Interior-point method, optimization approaches. There are also paper in which the mixture of such method types is applied. For example, in [56] the delay constrained primal problem is decomposed based on the levels of the cloud-fog architecture, and then the authors propose convex optimization, mixed-integer nonlinear programming and Hungarian method for the corresponding sub-problems.

Algorithm 1 Overview of the Embedding Algorithm [124] Returns a Set of Complete Mapping Structures of Service Graph S to Resource Graph R

```
 1: procedure MAP $(S, R) \to \mu, \lambda$
 2: $\Psi_S \leftarrow$ ORDERLEGSFORMAPPING$(V_S, E_S)$
 3: while $\exists e_j, j \in \Psi_S$ where $\nexists \mu(e_j, j)$ or $\nexists \lambda(e_j, j)$ do
 4:     while MAPONENF$(e_j, j)$ not successful do
 5:         $e_{j'}, j' \leftarrow$ GETBACKTRACKOPTION$(e_j, j)$
 6:         UNDOGREEDYMAPPING$(e_{j'}, j')$
 7:         $e_j, j \leftarrow e_{j'}, j'$
 8:     end while
 9: end while
10: return $\mu, \lambda$
11: end procedure
```

We grouped the rest of the applied techniques into the following method categories: Graph theory, Stochastic control, Machine learning, Evolutionary algorithms, Game theory, Matching theory, Linear algebra and Approximation. Interestingly, many of such techniques are applied to linear programming models, e.g., Shortest path and Best response search, Cache replacement strategies. For solving nonlinear and general programming problems such methods are proposed as Sine cosine algorithm, Constrained graph partitioning, Deep Reinforcement Learning and Genetic algorithms.

Naturally, the game theoretical and matching theoretical solving methods are also applied to problems defined in the respective framework, in Table VIII. Moreover, unconventional technique to problem pairings are also available in the related literature, such as using Eigendecomposition for Graph theory models, and Shortest path search to solve Optimal control problems.

For example, the MILP problem introduced by Formulation 1 is part of a hybrid, online-offline optimization framework in [171] and the offline optimization task (addressed by the MILP) is solved by Gurobi Mixed Integer Programming solver, whereas the online part is realized by a custom heuristic. Similarly, our other optimization example presented by Formulation 2 is solved by a dedicated heuristic algorithm in [124]. More specifically, the proposed orchestration engine runs a heuristic-guided greedy backtracking search on the resource graph structure. An overview on the formal description of this approach is shown in Algorithm 1. Refer to Table IX for a summary of the notations. An elementary mapping step of the algorithm is the greedy allocation of a VNF and an adjacent service graph link, called "leg" (e_j, j), onto a hosting (virtual) substrate node and path. An embedding order among these elementary steps is calculated by the function ORDERLEGSFORMAPPING(V_S, E_S).

In case a greedy step is not able to find a suitable host, while respecting all service graph requirements, the most recent greedy step is undone by freeing the temporarily reserved resources. $\textsc{MapOneNF}(e_j, j)$ is the core function realizing the greedy mapping of the leg while taking all the constraints into consideration. In each greedy step, the hosting substrate path and node pair with the lowest objective function value is chosen for mapping, and the next couple of best ones are stored for possible later exploration. The algorithm yields an embedding (μ, λ giving the hosting node of each VNF and the hosting path of each service graph link respectively), when all elements of the service graph have been successfully mapped respecting each aspect of the requirements or refuses the entire request. The search space size of the greedy backtracking can be tuned by the backtracking parameters (*i*) defining how many hosting alternatives of an elementary step shall be stored (branching factor), and (*ii*) how many consecutive greedy steps can be undone in the search tree (backtracking depth). The overall complexity of the embedding algorithm proposed in [124] including advanced features, such as end-to-end delay, node and link (anti-)affinity requirements is

$$\mathcal{O}\left(\max\left\{|V_R|^2|E_R| + |E_R|^2 + |V_S|, b\log b\right\}\frac{b^{k+1}}{b-1}\left\lceil\frac{|V_S|}{k}\right\rceil\right),$$

where *b* is the branching factor of the greedy backtrack search, while *k* is the backtracking limit, which shows the polynomial runtime in the input sizes $R = (V_R, E_R), S = (V_S, E_S)$ with low exponents.

We name a few examples here with the application field of the mathematical method. The authors of [69] define and solve a matching problem between BSs and UE. In [108], to solve the presented complex problem in polynomial time, the authors transform it into a more tractable form and then solve it by a sine cosine iterative algorithm. In [78] the applied k-means clustering not just determines the locations of, but also decides the number of fog nodes to be placed into the network. In [68], two-container cascade models are assumed, where the fog application is a cascade of a cloud and a fog module and the formulated mixed-integer nonlinear program is solved by an iterative greedy heuristic algorithm which is implemented and tested in the FogAtlas platform [186]. Another iterative heuristic is used in [146], where the fractional solution is rounded to a violating integer solution, which is gradually improved to find a good feasible solution. The scheduling is formalized as a shortest path problem in the graph of possible time slot configurations in [165], and it is solved by applying a dynamic programming approach. Evolutionary meta-heuristics are popular applied methods to solve placement problems [99], [119], [137], [138]. In [99], genetic algorithms are combined with Monte Carlo simulations for QoS optimizing, cost minimizing and utilization maximizing. Similarly, evolutionary algorithms are invoked in [119] to solve the VM placement problem in a federated cloud environment. The optimization problem is stated in a general format, consisting of embedding solutions, and the fitness function calculation is devised accordingly, defining all the necessary components of the genetic meta-heuristic. In [137], [138], the chromosome representation is a binary matrix, encoding the VNF to MEC node placements, while the fitness function is defined by the reciprocal of the solution costs [137], or a simulated annealing-based function of average end-to-end delay. The column generation method is used in [52], [53] to generate partial solutions, which improves the execution times of the large ILP and MILP formulations of the placement problems. Column generation requires close interaction with the ILP solver's logic, resulting in approximation algorithms with lower optimization times and performance guarantees. For more details, please consult Table VII.

C. Optimization Goals

Besides the modeling frameworks and the applied solving techniques, we also collect the optimization goals that show a heterogeneous picture. We depict the optimization goal(s) of each paper in the left-hand side of Tables XI and XII.

The end-to-end delay of the application to be deployed is the most widespread optimization goal, which is plausible, as decreasing the latency is the foremost purpose of edge computing platforms. Therefore the number of checkmarks is the highest in the leftmost column among the objective columns of Tables XI and XII.

The other three main objectives that occur among the optimization goals are energy, revenue and utilization, i.e., of resources of the system, are all intertwined optimization targets. Lower energy consumption in the fog infrastructure leads to less operational cost, hence higher revenue; similarly, if resource utilization is high, then larger revenue is generated from the hosted services, while the extent of capital expenditure, i.e., investment, in cloud infrastructure remains the same. Interestingly, there are papers that strive to minimize resource utilization, doing that with the ultimate goal of being capable of accommodating more future applications to come, hence higher revenue again on the long term. Nevertheless, we make the distinction between these three optimization objectives, and mark the ones addressed by each paper separately in Tables XI and XII. The energy target is predominantly set in mobile use cases, while the other objectives are general to both mobile and fixed communication infrastructure scenarios.

Other goals that are less frequently set and do not belong to any of the aforementioned four categories can be roughly classified in three groups. In the first group, networking aspects, other than latency, are optimized: either the throughput is maximized, or the bandwidth used by the placed tasks is minimized. In the second group, i.e., number of tasks and number of users, the goals are directly related to the utilization of the system, and directly or indirectly related to the revenue. Since both compute and network resources are relatively scarce and expensive in the edge, a few research papers handle the latter, e.g., network traffic, as optimization goals, rather than constraints. At last, there are the optimization goals related to some QoS aspects other than delay. Such an aspect is reliability, and the number of backups, that is strongly related to the resilience of the deployed service. Similar QoS consideration is availability, and all the related features and characteristics, such as load balancing, number of migrations and location.

TABLE XI
OBJECTIVES AND CONSTRAINTS OF THE OPTIMIZATION PROBLEMS ADDRESSED BY THE SURVEYED RESEARCH PAPERS (PART 1)

	Objectives					**Constraints**					
Papers	Delay	Energy	Revenue	Utilization	Other	Delay	BW	Computation	Radio	Mobility	Other
[49]	✓				migration	✓		✓			migration
[50]					bandwidth		✓	✓			
[51]	✓			✓			✓				reliability
[52]				✓	bandwidth	✓	✓	✓			
[53]					general cost		✓	✓			
[54]	✓					✓	✓	✓			
[55]			✓	✓	availability		✓	✓			reliability
[56]		✓				✓		✓	✓		
[57]			✓	✓		✓	✓	✓	✓		
[58]				✓		✓	✓	✓			
[59]			✓				✓	✓			cost
[60]		✓				✓	✓	✓			
[61]	✓			✓		✓	✓				
[62]	✓					✓	✓	✓			location
[63]				✓	#users			✓			coverage
[64]	✓								✓		#replicas
[65]					throughput	✓	✓	✓			
[66]					reliability						reliability
[67]	✓			✓	bandwidth	✓	✓	✓			
[68]			✓				✓	✓			storage, processing rate
[69]					throughput			✓	✓		
[70]	✓			✓	location			✓			
[71]	✓	✓			#tasks	✓		✓			
[72]	✓	✓	✓					✓		✓	
[73]	✓						✓	✓			location
[74]	✓					✓		✓			reliability
[75]		✓	✓			✓	✓	✓		✓	
[76]	✓				loadbalance	✓		✓			storage
[77]	✓							✓			location
[78]	✓										#nodes
[79]	✓				location		✓	✓		✓	
[80]	✓	✓						✓	✓		
[81]	✓	✓						✓	✓		energy
[82]				✓				✓			
[83]				✓	throughput		✓	✓			
[84]		✓				✓		✓			
[85]	✓	✓				✓		✓	✓		
[86]					#tasks	✓	✓	✓			storage, security
[87]	✓									✓	migration
[88]			✓				✓	✓			
[89]		✓			bandwidth			✓			location
[90]			✓					✓			reliability
[91]	✓						✓	✓			(anti-)affinity, processing rate
[92]				✓			✓	✓			storage
[93]			✓			✓	✓	✓	✓		
[94]	✓					✓					
[95]	✓		✓		reliability	✓	✓	✓			data amount
[96]					throughput		✓	✓			
[97]		✓		✓		✓	✓	✓			
[98]	✓									✓	migration
[99]	✓		✓	✓			✓	✓			
[100]	✓						✓	✓	✓	✓	
[101]	✓	✓					✓	✓	✓		reliability
[102]		✓				✓	✓	✓			security
[103]	✓	✓						✓			energy
[104]	✓							✓			
[105]		✓				✓		✓	✓		
[106]	✓		✓				✓	✓		✓	
[107]				✓		✓	✓	✓			user class
[108]				✓		✓		✓	✓	✓	reliability
[109](on)				✓		✓					
[109](off)			✓					✓	✓		
[110]	✓			✓				✓	✓	✓	energy
[111]			✓			✓	✓		✓		
[112]	✓	✓						✓	✓		
[113]			✓					✓	✓		
[114]			✓					✓			

It is important to emphasize that there are no primary and secondary objectives in the target functions of the related papers. Tables XI and XII depict tickmarks in multiple objective columns for cited papers that propose multi-variate optimization: usually a weighting factor is applied in the multi-variate function involved in the optimization.

TABLE XII
OBJECTIVES AND CONSTRAINTS OF THE OPTIMIZATION PROBLEMS ADDRESSED BY THE SURVEYED RESEARCH PAPERS (PART 2)

	Objectives					**Constraints**					
Papers	Delay	Energy	Revenue	Utilization	Other	Delay	BW	Computation	Radio	Mobility	Other
[115]	✓					✓		✓			reliability
[116]		✓	✓			✓		✓	✓		
[117]	✓	✓									
[118]	✓					✓	✓	✓			
[119]	✓		✓		location						general format
[120]			✓			✓	✓	✓			
[121]			✓			✓		✓			cost, location
[122]					loadbalance	✓	✓				
[123]	✓		✓	✓		✓	✓	✓			
[124]	✓		✓	✓		✓	✓	✓			(anti-)affinity
[125]	✓							✓			
[126]				✓			✓	✓			
[127]				✓		✓		✓			
[128]				✓			✓	✓			
[129]			✓	✓			✓				
[130]					#users			✓		✓	
[131]			✓	✓						✓	location
[132]	✓					✓	✓	✓			
[133]					throughput		✓	✓		✓	
[134]			✓			✓	✓				reliability
[135]	✓				loadbalance				✓		#users
[136]		✓	✓			✓		✓			
[137]					general cost		✓				storage
[138]	✓						✓	✓			
[139]				✓		✓		✓			location
[140]			✓	✓						✓	location
[142]	✓										energy
[143]	✓		✓				✓	✓	✓		
[144]			✓				✓	✓	✓		
[145]				✓	location	✓		✓		✓	
[146]					general cost	✓		✓		✓	battery, coverage
[147]	✓			✓				✓			
[148]	✓				migration					✓	location
[149]	✓	✓					✓		✓		processing rate
[150]	✓	✓				✓	✓		✓		processing rate
[151]	✓							✓			privacy
[152]			✓			✓	✓	✓			
[153]				✓		✓	✓	✓			processing rate
[154]		✓		✓			✓	✓	✓		
[155]			✓					✓	✓		
[156]	✓										
[157]	✓									✓	
[158]			✓				✓	✓	✓		
[159]	✓						✓		✓	✓	
[160]		✓				✓					
[161]	✓							✓			
[162]			✓			✓	✓	✓			
[163]			✓			✓	✓	✓			
[164]			✓			✓	✓	✓			
[165]					general cost		✓	✓		✓	migration
[166]				✓	#backups	✓		✓			reliability
[167]-1			✓			✓	✓	✓			
[167]-2	✓		✓	✓		✓	✓	✓			
[168]					#users	✓		✓		✓	
[169]	✓			✓	migration	✓		✓		✓	cost
[170]				✓	loadbalance	✓	✓	✓		✓	
[171]	✓			✓	bandwidth	✓	✓	✓			
[172]			✓			✓	✓	✓			VM states
[173]			✓							✓	coverage
[174]			✓			✓	✓	✓		✓	
[175]				✓		✓		✓			
[176]				✓			✓	✓			storage
[177]	✓		✓			✓	✓	✓			
[178]				✓			✓	✓			storage
[179]	✓		✓			✓	✓	✓			anti-affinity
[180]			✓			✓	✓	✓		✓	
[181]	✓		✓					✓		✓	energy
[182]			✓		availability			✓		✓	migration
[183]	✓	✓									privacy

For example, in the presented MILP problem (Formulation 1), the objective function is defined by Equation (1). As we have seen, the first term represents the costs for allocating bandwidth on given links, while the second term describes the costs for mapping a service node onto a resource node, and finally the last term expresses

Algorithm 2 Details of the Objective Value Calculation for Greedily Choosing the Locally Most Preferred Resource Node and Hosting Path for a Leg. Reference [124] Returns a Real Value, Which Is Used to Sort the Hosting Options of a Leg

```
• $\Omega_x$, $\rho_x$ are the value and weights of the bandwidth, resource, latency, cost components, denoted by $x \in \{bw, res, lat, cost\}$ respectively.
• $\omega_r$ are the weights of node resource component $r$.
• $\xi_1$, $\xi_2$ are the weights of the latency components.
1: procedure CALCOBJECTIVEVALUE($e_j, j, p_{u \rightsquigarrow v}, v$)
2:     $\Omega_{bw} \leftarrow$ GETAVERAGEPATHBWUTIL($p_{u \rightsquigarrow v}$)
3:     $\Omega_{res} \leftarrow \sum_{r \in \mathcal{R}} \omega_r$GETNODERESUTIL($v, r$)
4:     $\Omega_{lat} \leftarrow \xi_1$DISTANCEFROMLASTHOST($p_{u \rightsquigarrow v}$) + $\xi_2$DIRECTTOWARDSENDOFPATHLATENCY($e_j, j, v, \mu$)
5:     $\Omega_{cost} \leftarrow$ GETCOSTOFLEGHOST($p_{u \rightsquigarrow v}, v, e_j, j$)
6:     return $\sum_{x \in \{bw,res,lat,cost\}} \rho_x \Omega_x$
7: end procedure
```

the extent of load balancing in the substrate network. The goal is to find the allocation with the minimal cost assuming predefined weighting factors for the constituent terms. (It is worth noting that the delay is not part of this objective function, however, the online part of the hybrid optimization framework proposed in [171] optimizes for the delay as well in its heuristic. That is the reason for the corresponding checkmark in Table XI.)

The optimization problem from [124], which is summarized by Formulation 2, includes the general objective function formulated in Equation (17), which minimizes the sum of the objective value of each VNF $j \in V_S$ and the adjacent service graph connection $e_j \in E_S$. This objective function is invoked in each greedy step by Algorithm 1 in order to find the hosting substrate path and node pair with the lowest objective function value. The details of how the embedding possibilities are sorted is shown in Algorithm 2. Besides the resource availability and greedy search directing objective function components, GETCOSTOFLEGHOST($p_{u \rightsquigarrow v}, v, e_j, j$) calculates the cost of using a hosting resource graph path $p_{u \rightsquigarrow v} \in \mathcal{P}(E_R)$ and host $v \in V_R$ for (e_j, j). A given setting for the weights of latency subcomponents ξ_i, resource type subcomponents ω_r and objective components ρ_x provide a fully specified optimization goal for the mathematical problem statement in Equation (17). By tuning these weights, operators can adapt the algorithm to multiple application scenarios. By these means, a versatile multi-domain orchestrator can be implemented which is capable of: (*i*) optimizing for bandwidth utilization on infrastructure connections, (*ii*) distributing node resource utilization among operators, (*iii*) providing high service acceptance for delay critical applications, (*iv*) minimizing administrative costs of routing and VNF hosting, or (*v*) arbitrary superposition of multiple operating policies from the various involved entities.

D. Optimization Constraints

Analogue to the duality in optimization forms, the set of constraints assumed in the papers is more or less the same as the set of objectives. We list the five major groups of constraints in the right-hand side half of Tables XI and XII, and we sign with a checkmark if the given paper considers the constraint in its optimization formula. In case other types of constraints are also included in the paper, we denote them in the rightmost column of said tables.

The delay is not the mostly seen constraint throughout the collected papers, because in the majority of them, the delay is (a part of) the objective to minimize. In those papers where it is not, delay is usually considered as a constraint. The most frequently applied constraint, observed in almost all the collected papers, is on computation resources. Moreover, besides the delay, which is required by latency-sensitive applications, and the computation, e.g., CPU, memory, which is dictated by the capacity of the underlying infrastructure, there are constraints formalized on network resources, i.e., network bandwidth, denoted by BW in Tables XI and XII. This latter belongs to these three most important constraints, because particularly in edge networks, bandwidth is often assumed to be a scarce resource.

As the emergence of MEC induced an important body of research effort in efficiently handling wireless networks, radio spectrum resources are also considered as constraints in a number of papers. Again, due to the field of MEC, user mobility can also be taken into account in the optimization as guiding constraints. Surprisingly, there are papers in which researchers formulate constraints not only on the mobility of users, but also on the mobility of edge nodes, e.g., that are mounted on UAVs. Therefore the radio and mobility constraints are predominantly dictated in mobile use cases, while the other constraint dimensions are general to both mobile and fixed communication infrastructure scenarios.

Other, less frequently seen constraints, just like in the case of objectives, include cost and reliability, but other types of constraints are applied as well. We identify four groups into which these occasionally seen constraints can be classified. One can come across most frequently with research works that build constraints for sustaining the reliability aspect of service QoS. We denoted those constraints in this group with tags of reliability, number of replicas and number of nodes. The second most prevalent type is about location: either the location constraints given by the infrastructure, e.g., tags of location and coverage, or the location of deployed service components, e.g., tags of migration and affinity. We assign constraints about costs into the third category: those papers that define cost-related constraints are tagged with cost or energy labels in Tables XI and XII. Somewhat connected to the computation resources, we identify more explicit constraints, such as processing rate and storage with these tags in the rightmost column. Finally, a category of constraints that is never considered as optimization objective revolves around security: the tags security, privacy, and user class are applied to those papers that take into account user and data security aspects.

Similarly to our note for optimization targets, we underline the fact that there is no priority distinction between constraints, i.e., there are no primary and secondary constraints in the related work. However, the predominant majority of cited work in Tables XI and XII introduce multi-constrained optimization problems, hence multiple tickmarks in each row within the Constraints part of the tables.

TABLE XIII
GENERAL NOTATION USED TO UNIFORMLY DESCRIBE THE COMPLEXITY ANALYSIS RESULTS OF THE SURVEYED PAPERS

Notation	Description
N	Number of possible placement options in the substrate topology, such as BSs, cloudlets, edge/fog nodes, MEC servers, private/public clouds or other infrastructure nodes.
E	Number of edges or links in the substrate topology.
S	Number of service components or units to be placed, such as applications, micro-services or parts of it as containers, pods or VMs, as well as requests, tasks, jobs or functions.
L	Number of logical links between service components.
U	Number of users or UEs.
T	Number of time slots or time periods to be considered or the size of a specific look-ahead window.
k	Number of iterations for internally generated scenarios, such as sub-problems, random samples or branching factor.
m	Maximal length of a chain in the service.
r	Number of replica copies or replication factor.
c	Number of sub-channels or sub-bands.
ξ	Convergence threshold/limit or solution accuracy.
$\mathcal{LP}$	Computational steps required by a Linear Program solver.

As an example, the MILP problem from [171], presented in Formulation 1, formulates the constraints in equations (2)-(10). As we have discussed in Section VI-A, Constraints (2), (3), (5), (8) define the constraints related to computation resources, while Constraints (4), (6), (9) control bandwidth allocations on substrate links. The end-to-end latency constraints are formulated by Constraint (10) for respective service function chains. Our other example from [124], highlighted by Formulation 2, includes similar constraints formulated by Constraints (13)-(16). The only difference is the distinction between link-wise delay requirements, described by Constraint 14, and end-to-end latency constraints, formulated by Constraint 16.

E. Complexity Analysis

In this section, we analyze the algorithmic complexity of the proposed methods within the collected research papers. From the set of the articles, we selected those that explicitly addressed complexity or made a clear statement about the complexity of the published optimization method therein. For the sake of comparability, we introduce a common, unified notation summarized in Table XIII. The overview of these selected papers is depicted in Table XIV. The contents of the table reflect the exact and final complexity analysis results of the cited papers: we have not extended their statements in any way, e.g., we do not combine their partial results if the authors did not do so in their article.

After careful evaluation of the heterogeneity of complexity analysis approaches within the selected papers, we group the papers into three formalization categories and depict each category in the Form column of Table XIV. The categories are the following.

O: the final complexity of the algorithmic solution proposed for the placement problem is given in closed form by an explicitly defined asymptotic notation;

T: the analysis is provided in a textual formalization, e.g., the algorithm's total complexity is specified typically to be polynomial or exponential, while only the complexity of sub-steps is given by asymptotic notations;

E: the complexity is indirectly derived from other external algorithm's complexity, e.g., that of an LP solver.

In all categories, we use a unified notation for ease of comparison; the unified notation is introduced in Table XIII. The majority of formulas contain the number of servers or nodes (depicted by N), the number of service components (depicted by S), and the number of users (denoted by U) as inputs. We grouped the references in Table XIV based on their proposed Service structure types, given in the two leftmost columns. Within each group, the articles are sorted based on the complexity (in terms of *N*) in the rightmost column from the simplest to the most complex.

For the papers that fall into the category Form O, all the parameters included in the complexity formula are given either in Table XIII (common parameters) or in Table XIV (special parameters). All these papers provide a formal description of their proposed algorithms' complexity, therefore the comparison of their expected runtime is straightforward. Among the papers that fall in the category Form T, authors typically propose iterative solutions. Several papers provide the complexity of only one iteration step, either for simplicity and brevity [89] or because the number of iterations in the algorithm required to arrive at a solution depends on the objective function's curvature that is difficult to quantify [101]. Alternatively, other papers do reveal the number of iterations, but not the complexity of all steps [54]; or only certain substeps are analyzed [75], [81], [111]. In the third category denoted by Form E, one sub-step is typically a linear programming method: [93] builds on the polynomial-time solvability of LP problems, whereas [126], [128], [133] explicitly rely on the complexity of the Simplex method. Furthermore, the total complexity shown in [178] is based solely on an LP solver.

We have taken into account the selected papers' optimization objectives and constraints from Tables XI and XII in order to distill meaningful observations on their effect on the complexities of the proposed algorithms. To sum up the relation between the optimization tasks and their complexities, we conclude the following. In case of offloading problems and problems addressing single component type applications, the more the terms that are considered in the objective function and in the constraints, the more complex algorithms with a larger number of steps are required. Similar conclusion cannot be drawn for other types of tasks. In general, when the utilization is included in the objective function, the proposed algorithms become more complex; this suggests that the approximation of the underlying problem, i.e., optimizing the utilization, requires more sophisticated heuristics. In addition, taking the bandwidth, throughput or load balancing into account in the objective function, the algorithms usually result in longer runtime. And finally, according to the surveyed results, there is no direct relation between the exact constraints and the complexity of the proposed algorithms.

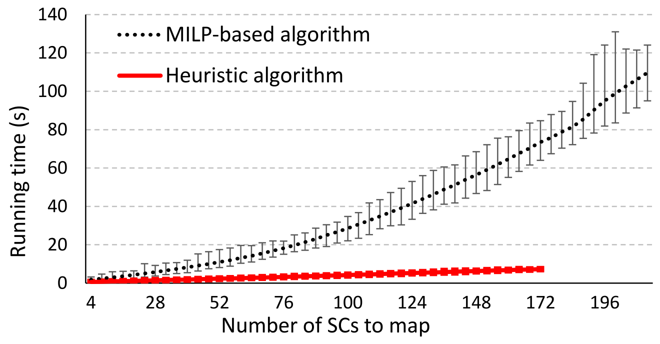
Unfortunately, the accuracy of the proposed heuristic algorithms are rarely touched upon, therefore we are not able to draw a comprehensive analysis in this aspect. However, it

TABLE XIV
COMPLEXITY OF THE PROPOSED ALGORITHMS. O: CLOSED FORM; T: DESCRIBED IN A HIGH LEVEL, WHILE ONLY SUB-COMPONENTS OF THE PROPOSED SOLUTION ARE CHARACTERIZED BY ASYMPTOTIC NOTATION; E: DERIVED FROM EXTERNAL ALGORITHM'S COMPLEXITY, E.G., AN LP SOLVER'S

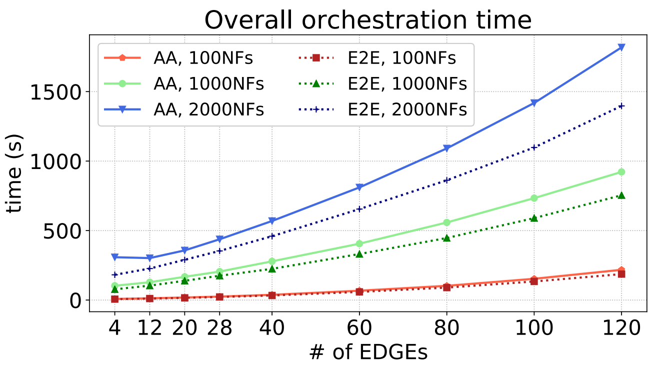
Service structure		**Paper**	**Complexity of the proposed method**	
Components	**Task Type**		**Form**	**Computational Complexity**
Single component	Offloading	[113]	O	$\mathcal{O}(U^3)$
		[81]	T	The algorithm's complexity is proved to be linear in U, N, c.
		[112]	O	$\mathcal{O}(\lceil\log_2(P/\xi)\rceil\rho(U,N,c)\log\delta)$, where P is the maximum transmission power, ρ is a suitably chosen polynomial and δ is the optimal/initial ratio.
		[117]	O	$\mathcal{O}((S(N+1))^4(S(N+1)+N+S+2)^{0.5}\log(1/\xi)+kNS)$
		[101]	T	The algorithm's complexity is concluded to be polynomial in the number of U.
	Edge/fog computing	[165]	O	$\mathcal{O}(Sk^2T)$
		[86]	O	$\mathcal{O}(SN)$
		[130]	O	$\mathcal{O}(UN)$
		[90]	O	$\mathcal{O}(S\log S+SN)$
		[93]	E	$\mathcal{O}(SN*\mathcal{LP})$
		[82]	O	$\mathcal{O}(N\log N)$
		[98]	O	$\mathcal{O}(N^2T)$
		[181]	O	$\mathcal{O}(NU^2+UN^2T^2k)$
		[69]	O	$\mathcal{O}(N^2U^2S)$
		[166]	O	$\mathcal{O}(S^2N^2+N^3)$
		[177]	T	The algorithm's complexity is proved to be polynomial in N, S.
Multiple components	Offloading	[114]	O	$\mathcal{O}(S^{3.5}N^{7.5}U^2+N^{7.5}U^{5.5})$
	Edge/fog computing	[182]	O	$\mathcal{O}(\eta(U+N+2\delta)+k\eta\delta+\delta)$, where δ,η are particle parameters of the PSO.
		[74]	O	$\mathcal{O}(NS)$
		[70]	O	$\mathcal{O}(3SN)$
		[129]	O	$\mathcal{O}(S^2Nd)$, where d is the diameter of the communication network.
		[169]	O	$\mathcal{O}(S^2N^2)$
		[68]	O	$\mathcal{O}(S^2N^2+S^2)$
		[133]	E	$\mathcal{O}((N+B)(SN*\mathcal{LP}))$, where B is the number of special backhaul nodes.
		[75]	T	The proposed solution's complexity is polynomial in N, S.
		[131], [140]	O	$\mathcal{O}(N^3+NS^3)$
		[64]	O	$O(S^2N^3r)$
		[59]	O	$\mathcal{O}(N^{13}S^{6.5})$
		[176], [178]	E	The approximation can be solved in polynomial time using an $\mathcal{LP}$ solver.
		[127]	T	The algorithm's complexity is exponential in the number of service chains.
Multiple connected components	Offloading	[109](on)	O	$\mathcal{O}(S^2)$
		[109](off)	O	$\mathcal{O}(U\log U)$
		[111]	T	The algorithm's complexity is proved to be linear in N, S.
		[145]	O	$\mathcal{O}(N)$
		[139]	O	$\mathcal{O}(S^2N)$
		[107]	O	$\mathcal{O}(S^2N^2)$
	Multi-cloud	[172]	O	$\mathcal{O}(viSL\log L+i\log i)$, where v and i are the number of instances of the same VNF type and the number of invocations w.r.t. a given service.
		[118]	O	$\mathcal{O}(N)$
		[124]	O	$\mathcal{O}(\max\{N^2E+E^2+S,k\log k\}\frac{k^{\xi+1}}{k-1}\lceil\frac{S}{\xi}\rceil)$
		[122]	O	$\mathcal{O}(S^3N^{(2+h)})$, where h is the maximal junction node count in the requests.
		[50]	O	$\mathcal{O}(N^3)$
		[53]	T	Fixed-parameter tractable in the request graph treewidth.
	Edge/fog computing	[134]	O	$\mathcal{O}(S^2\log N)$
		[65]	O	$\mathcal{O}(SNEm)$
		[83]	O	$\mathcal{O}(SN\log N)$
		[76]	O	$\mathcal{O}(S^2N\log N)$
		[66]	O	$\mathcal{O}(E+N(\log N+r))$
		[61]	O	$\mathcal{O}((N+S+NS)\log N+S\log S)$
		[92]	O	$\mathcal{O}(SL(N\log N+E))$
		[67]	O	$\mathcal{O}(N^2)$
		[97]	O	$\mathcal{O}(kN(k\log_2 N+S\log_2 S)+N(m+N\log N))$
		[146]	O	$\mathcal{O}(S^5N^4T)$
		[126], [128]	E	$\mathcal{O}(N^4*\mathcal{LP})$
		[89]	T	The algorithm's complexity is proved to be polynomial in N, S.
		[162]	T	The computational complexity of the algorithm is stated to be polynomial.

would be interesting to see how the approximation bound of the proposed heuristics relate to their algorithmic complexities.

In order to illustrate the applicability of the surveyed methods, we present two experimental results corresponding to our explanatory examples (MILP and VNE) presented in this section. In Figure 11(a), a preliminary version of the MILP-based solution (from [123]) is compared to the basic variant of the heuristic VNE algorithm proposed in [124]. (Here, the basic



(a) Comparison of the runtimes of the MILP-based algorithm proposed in [123] (which is a preliminary version of the solution given in [171] and presented in this section) and a basic version of the heuristic VNE algorithm provided in [124] and briefly summarized in this section.



(b) Scalability of the advanced heuristic VNE algorithm proposed in [124] with resource domains (number of managed edge sites). Two types of advanced constraints were configured in the service requests: end-to-end delay (E2E) and anti-affinity (AA) between dedicated VNF pairs.

Fig. 11. Illustrative results of selected control plane experiments on emulated edge/cloud infrastructures published in [123] and [124].

operation means that the advanced features related to service reliability are not used.)

Both placement methods have been evaluated on a real world topology taken from SNDlib [187], which has 42 nodes and 157 edges, representing access, aggregation and core network parts, equipped with computation resources. The `dfn-gwin` topology was used with additional network parts and computing resources. All computation nodes had the same available resources, i.e., 400 units of CPU each. Service requests were service chains (SCs) consisting of 1 to 8 VNFs with different resource requests generated following a uniform distribution. In order to compare the capabilities of the two approaches, we grouped a sequence of service chains together into one batch request and that was given as input for both algorithms. It is worth noting, that the heuristic VNE algorithm can operate either in online mode (when the input sequence is received and processed step-by-step) or in offline mode (when the overall request is known in advance). The simulation was conducted on computers with Intel Core i5 processors and 8 GB RAM. The results are shown in Figure 11(a), where the error bars represent the minimal and maximal runtimes among the 100 independent service chain sequences. On the one hand, the heuristic algorithm exhibits polynomial scaling with the number of input service chains, while the MILP-based approach imposes impractical runtimes and exhibits worse scaling behavior. On the other hand, according to [123], the heuristic algorithm can embed around the $(2/3)$ (varying between 59-75%) of the optimal, offline calculated number of requests. When we are close to a saturated state, the heuristic method cannot map the overall request, this is the reason why we do not have samples above 172 chains regarding the heuristic solution.

The advanced version of the VNE algorithm provided in [124] was also evaluated on different edge/cloud scenarios. Selected results are shown in Figure 11(b). Here, a core full-mesh network was used to connect 4 cloud servers providing the majority of the available computational resources and the distant edge nodes that were capable of running a limited number of VNFs. A dedicated orchestrator instance was in charge of managing all involved domains. By these means, a single flat control plane hierarchy was established. Figure 11(b) illustrates the impact of the number of simultaneously controlled edge domains tested with different size of 5G-aware services. Two types of service requests were evaluated: one with end-to-end delay requirements comparable to the diameter of the topology and another one containing anti-affinity constraints between dedicated VNF pairs. The overall runtimes of these services are comparable to each other and show that services with anti-affinity relations require slightly more time for the orchestrator to deploy. According to [124], the evaluation confirmed the polynomial scaling properties of the orchestration system in terms of both the infrastructure size and the complexity of the service requests. Assuming a reasonable number of edge sites and service requests including less than 100 VNFs, the placement calculation takes only a few seconds. When we have more than 1000 service elements in the request (which is not typical today), the orchestration time takes a few minutes. And finally, when the number of edge domains is also large (e.g., around 100), the orchestration time can take almost half an hour. However, in terms of management time scale, it is not considered as an extreme long duration and it can be acceptable in certain scenarios.

VII. Research Gaps

After careful evaluation of the spectrum of research results in cloud-edge placement, we consider the following interesting topics yet to be investigated. We argue that all the listed aspects are of utmost importance for an edge computing platform to fully serve the edge applications' and their users' needs, furthermore all of those greatly affect the placement logic that must be deployed in the system.

A. Awareness to User Mobility in Edge Placement Decisions

Combining classical user mobility models, of which the investigation started in the previous millennium, e.g., [188], and modern AI-based predictions, e.g., [189], locality-aware, proactive hybrid service placement techniques might be a promising area for further improving service QoS and effectiveness of resource provisioning. We have seen several papers taking user (or host) mobility into account in their optimization problems' constraints (see the second rightmost column of Tables XI and XII), but using AI models for accurate prediction of user movements (if patterns can be found) could significantly enhance the reachable optimization target in dynamic systems that allow for migration of service components and/or complete re-optimization of placements periodically. Closely related to mobility aspects, interestingly no related work has considered, to the best of our knowledge, using smart cars as edge nodes. We argue that with the user mobility patterns in hand, the driving habits could be exploited: the predicted paths and locations of electric cars could serve as an input to the dynamic planning of edge infrastructure if one assumes that the computing power of those cars can be utilized, especially during charging, e.g., at a shopping mall.

B. Cross-Layer Optimization With Edge Application Design

We have found that although sophisticated placement decision making is often implemented in cloud-edge management frameworks, an API is rarely exposed to the application owner or software developer about the placement policy in operation. This could be however beneficial, beyond an API functionality that merely supports stating the application requirements, in order to develop the applications against it with a larger extent. An obvious example would be the disclosure of the optimization target function and a query possibility for the overall system status, in order to adapt the application-level parameters according to the expected placement decisions. In general, a deeper integration of cloud-native applications' behavior with the cloud platform's management policy might lead to a more efficient operation.

C. Joint Placement of Functions and Their Corresponding Data

The computational entities that are placed in the cloud-edge infrastructure often belong to heavy or light virtualization technologies: VMs and containers, respectively. In several papers big data processing tasks are also orchestrated, but in general, the latest trend of Function-as-a-Service (FaaS), also called as serverless, platforms are rarely addressed. As those systems usually accommodate ephemeral functions, their load is hectic, and hence the placement logic to be applied must be fast and efficient. Function requests arrive frequently, demand various resource amounts and a great level of elasticity. Therefore, by its nature the FaaS service platform must apply an online placement method, probably with affinity constraints to consider in order to collocate functions that may invoke each other [190]. Inherent to the FaaS concept, input data or internal function state are often externalized, hence the stateless operation. As network delay might cause serious QoS degradation when remote data must be accessed by the functions invoked in the FaaS platform, the placement of those is of paramount importance too [190]–[192]. We advocate the emergence of joint placement policies of functions and their respective states in edge systems in the near future.

Coded Distributed Computation [23] is a new research field that has emerged in recent years, where the function placement is strongly related to the input data allocation. CDC is a mixture of coding techniques and distributed computing to reduce the communication costs of the large-scale computation of tasks and to mitigate the straggler effects in a distributed environment. Although both data and function allocations are essential elements of the CDC scheme, most of the studies focus on only one of them. E.g., the work presented in [193] about optimal file allocation considers uniformly distributed processing functions among the cluster nodes. In contrast, authors of [194] present a Reduce function placement while they assume equally split input files among the nodes. Since data reallocation could easily result in high communication load in the cluster, we argue that the joint placement optimization of both functions and their input data is essential.

D. Temporal Placement Policies for Auto-Scaling Edge Applications

The integration of placement methods with the auto-scaling capabilities of widely used platforms, e.g., Kubernetes [195], [196], or with reliability measures [166] necessary in an edge infrastructure that is prone to errors and downtimes, might be a challenging research avenue. Indeed, the optimization problem is rendered to be highly complex by integrating the dynamics of the deployed services and the actions the platform takes in turn. As many targeted use cases require end-to-end latency guarantees, the emergence of real-time cloud, and particularly, that of the real-time edge is imperative.

E. Security-Aware Edge Placement Challenges

Unfortunately security aspects receive little attention in the body of research on edge platforms and placement techniques. The authors of [197] argue that it is the hasty design and development of these systems that has led to the neglect of security threats in the edge computing platforms themselves and in their enabled applications. The rare appearance of security and privacy in the rightmost column of Tables XI and XII proves their point. In recent surveys [197], [198] one may learn about basic attacks, as well as the corresponding defense mechanisms, furthermore about the latest research advance of data security and privacy-preserving protection technologies, specifically tailored to the field of edge computing.

A stellar example of research results in the area of privacy [199] has tackled the challenge of improving the overall execution performance of edge computing nodes, i.e., the resource usage, the load balance levels, and the power consumption, while preventing privacy leakage of the IoT devices for service placement. The authors propose a trust-oriented IoT service placement method for smart cities in edge computing that provides balanced placement strategies for the trade-offs among the execution performance metrics with privacy preservation. Their example urges the research community to consider such security and privacy requirements when designing the placement engines in the core of edge platforms, as we see that by the commoditization of edge computing services such features will be mandatory in order to maintain the trust of users. E.g., the authors of [200] and [201] assume that edge nodes are not trustworthy so they propose to apply differential privacy manipulations on user data before sending those to the cloud; we argue that their assumption holds even stronger in a public edge multi-provider setup, however advanced orchestration techniques might mitigate the problem, e.g., by data obfuscation over multiple edge nodes of multiple edge service providers.

F. Economic Aspects of Edge Placement

The economics of placement decisions in edge platforms are often incorporated into the orchestration logic in related work. The collected papers associated with tick marks in the Revenue optimization goal column of Tables XI and XII explicitly account for the maximization of profit generated by edge computing service. Revenue maximization may be performed by effective resource usage and by admitting more customers to the service, alternatively the applied pricing scheme can be tailored to find the sweet spot in the income and cost trade-off. Specifically, the interaction between users and an edge cloud provider through the pricing of the computation offloading service has been targeted by many research endeavors. E.g., in [202] the authors consider two important metrics: latency and fee, and they formulate a stochastic game to model the interaction between users and the provider. In this game, the provider sets prices to maximize its profit, while users devise the offloading strategy to reduce both the latency and charge.

Nevertheless, the emerging technology of serverless computing has recently inspired researchers to consider the edge placement problem from a novel perspective. In a serverless platform users are able to deploy individual functions and pay only for the time that their code is actually executing. The pricing model usually depends on the memory, duration, and the number of executions of a sequence/workflow of functions. The authors of [203] adapt the cloud native approach and related operating techniques for latency sensitive IoT applications operated on public serverless platforms. They argue that solely adding cloud resources to the edge is not enough and other mechanisms and operation layers are required to achieve the desired level of quality. Therefore they propose a novel system on top of a public serverless edge cloud platform, which can dynamically optimize and deploy the microservice based software layout based on live performance measurements. They apply their concepts to one of today's most widely used and versatile public cloud platforms, Amazon's AWS, and its edge extension for IoT applications, called Greengrass. Also considering Amazon's serverless offering, AWS Lambda, the authors of [204] present an algorithm that optimizes the price of serverless applications by, among other methods, splitting functions across edge and cloud resources, and allocating the memory for each function. They present an efficient algorithm to explore different function placement solutions and find the solution that optimizes the application's price while keeping the latency under a certain threshold. In general, we still see a research gap in the area of cost optimization of application owners. The complex optimization we envision overarches the cloud application's whole lifetime: starting from the application design, e.g., monolith or microservice-type implementation, through the choice of cloud services, e.g., VM or container or function as a service, to the selection of the ideal public cloud provider, both in terms of pricing and service quality, e.g., locality and performance, to deploy the application. The placement of computational units is in the focal point of this complex challenge, raising hard computational problems to solve.

G. Multi-Operator Setting

We identify the lack of the multi-operator scenario as the most common shortcoming in the related work results. The vast of the research works assume a dedicated central entity which is in charge of calculating the efficient placement of the service components. Those researchers assume single-operator scenarios, i.e., the resources operated in central clouds or deployed to edge domains are owned and managed by a single provider. This seems to be a realistic assumption and the first generation of edge and fog computing infrastructures comply with this model. However, envisioned large scale, e.g., global scale, service deployments require compute and network resources from multiple providers and only a federation of operators can provision these services and applications. Unfortunately, only a few works [124], [167] address the more complex and more challenging multi-operator setups. Modeling such a scenario, and combining the optimization problem with the economical aspects described in Section VII-F would create exciting research problems and valuable results.

VIII. Summary

Recent cloud architectures, proposing the deployment of computation resources to the edge of the network, enable a new generation of network services and applications. However, simply extending the cloud per se is not enough and novel features, capabilities and workflows should be added. As we have seen, a key component which significantly affects the application's performance, especially if it is constructed from multiple communicating modules, is the algorithm responsible for calculating the optimal placement of computational units (varying from VMs to fine-granular software functions). As an answer, the research community has dedicated significant efforts to this challenging topic and a vast number of theoretical papers have been published during the recent years. This survey paper aimed at categorizing the current results related to the placement problem in the edge. A structured taxonomy has been defined and the surveyed solutions have been presented according to the identified dimensions considered relevant. Starting from the aspects of the underlying cloud architecture and the structure of the supported services, across the dimensions of the mathematical tool set and the applied methods, we arrived at the detailed characterization of the explored optimization problems including the targeted goals and considered constraints, which of course determine the family of supported use cases, as well.

The vast body of literature on the placement of edge applications offers a colorful combination of use cases, problem formulation and selected methods applied to solve the problem. An imaginary paper that we would place in the focal point of the multi-dimensional space of the categorization characteristics we identified in this survey would propose a framework to push the resource-intensive applications from end devices to the edge, and delay critical components from the cloud to the edge. This average paper would propose an online algorithm to do so with the goals of minimizing average data traffic in the edge in order to save battery in the end devices. The authors would decompose the original problem into a resource allocation problem with fixed task offloading decisions, and a task offloading problem that optimizes the value function corresponding to the resource allocation problem; separate sub-problems would then be formulated as ILP, and solved by proposed heuristics. One distinctive feature of the edge system would be added, e.g., to provide latency constrained service access for customers, services would be replicated from the cloud to a selected subset of the edge servers.

Of course, each and every paper we collected for this survey is special on its own. However, as a general summary, we argue that the goals of the researchers are similar. In MEC offloading papers the goal is selected from a rather limited set of inherent choices: energy and processing time. End users and/or terminals aim at minimizing their own battery usage (induced by the low power nature of IoT devices in general) or the processing time that constitutes the service latency. Extending this goal concept, several research initiatives have been made to tackle the trade-off between power consumption and transmission delay. Others focus on resource allocation in edge and fog computing systems, mainly considering cost and/or delay as optimization objective(s). In general, cost minimization is a widely applied formalization in the research on edge computing platforms, and the cost can stand for different aspects, either resource consumption (i.e., efficient utilization) or service quality and its related revenue. Minimizing the perceived latency or respecting a delay constraint is linked to the revenue from the operator's point of view.

We find the objects to be placed are the second main point in most of the collected papers: usually those are application instances, or the components thereof. However, some papers combine the optimization of the virtual infrastructure elements with the placement of service components. A few others, instead of directly placing applications, model the workload as individual tasks: within these models the service requests are single entities to be placed onto a heterogeneous edge computing network. Naturally, the infrastructure itself is closely related to the objects to be placed: when the application contains non time-critical components, then central cloud is an available choice, otherwise the alternatives are edge nodes and end devices.

The third common aspect in the related work is mobility. One can find papers in the related literature that apply moving devices, e.g., UAVs, robots, as edge nodes in order to position them to optimal locations anytime the user terminals change their location. Migration-enabled and mobility-aware approaches, in which user migration is considered between adjacent base stations, aim to maximize the user coverage rate, to minimize the number of re-allocations and to yield refinable dynamic allocations. Therefore in several research papers mobility patterns are incorporated in the models and the service deployment and/or resource allocation are optimized either proactively based on predictions or reactively based on measurements.

Overall we find this survey to be extremely useful for researchers, engineers, system designers, and developers for several reasons. First, the summarized mathematical tool set of the collected papers empowers the reader with a good understanding of potential formal approaches to related future problems from other research fields. The pros/cons of different modeling and problem solving frameworks thus can be identified in advance based on the lessons learned this survey conveys. The suitable methods a scholar can select from for their respective research agenda is offered as a comprehensive, but bounded search space. Therefore, the design of algorithms for related problems will be easier with all the pointers to the technical details of promising solutions. Second, as a more use case oriented fruit of this work, this survey delivers a comprehensive catalog on the mathematical apparatus for the placement problem, which is essentially a tutorial guide for applied research and product development. For example, the design and implementation of an orchestrator software of an arbitrary edge cloud platform or service can be driven by selecting the most suitable solutions presented in the easily searchable taxonomy structure. As an addition, the algorithmic complexity characteristics are also provided, so the practical feasibility of the selected approaches can be instantly evaluated. Third, the disclosed research gaps list promising future research directions for those scholars that are active in the topic of edge cloud scheduling and orchestration in challenging setups.

REFERENCES

[1] *Amazon Web Services*. Accessed: Oct. 1, 2020. [Online]. Available: https://aws.amazon.com

[2] *Google Cloud Platform*. Accessed: Oct. 1, 2020. [Online]. Available: https://cloud.google.com

[3] *Microsoft Azure*. Accessed: Oct. 1, 2020. [Online]. Available: https://azure.microsoft.com

[4] *OpenStack*. Accessed: Oct. 1, 2020. [Online]. Available: https://www.openstack.org

[5] *Docker: A Better Way to Build Apps*. Accessed: Oct. 1, 2020. [Online]. Available: https://www.docker.com

[6] *Kubernetes: Production-Grade Container Orchestration*. Accessed: Oct. 1, 2020. [Online]. Available: https://kubernetes.io

[7] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surveys*, vol. 52, no. 6, pp. 1–36, Oct. 2019.

[8] 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.

[9] 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.

[10] A. Yousefpour *et al.*, "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.

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

[12] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1826–1857, 3rd Quart., 2018.

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

[14] Z. Zhao, G. Min, W. Gao, Y. Wu, H. Duan, and Q. Ni, "Deploying edge computing nodes for large-scale IoT: A diversity aware approach," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3606–3614, Oct. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8333700/

[15] H. D. Chantre and N. L. S. Da Fonseca, "The location problem for the provisioning of protected slices in NFV-based MEC infrastructure," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 7, pp. 1505–1514, Jul. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9062537/

[16] R. A. C. da Silva and N. L. S. da Fonseca, "On the location of fog nodes in fog-cloud infrastructures," *Sensors*, vol. 19, no. 11, p. 2445, May 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/11/2445

[17] R. A. C. da Silva and N. L. S. da Fonseca, "Location of fog nodes for reduction of energy consumption of end-user devices," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 593–605, Jun. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9062311/

[18] M. T. Covas, C. A. Silva, and L. C. Dias, "Multicriteria decision analysis for sustainable data centers location," *Int. Trans. Oper. Res.*, vol. 20, no. 3, pp. 269–299, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2012.00874.x

[19] L. Toka, B. Gema, and B. Sonkoly, "A stable matching method for cloud scheduling," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–6.

[20] F. Meneses, M. Fernandes, D. Corujo, and R. L. Aguiar, "SliMANO: An expandable framework for the management and orchestration of end-to-end network slices," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–6.

[21] Z. Sun, P. Du, A. Nakao, L. Zhong, and R. Onishi, "Building dynamic mapping with CUPS for next generation automotive edge computing," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–6.

[22] F. Jalali *et al.*, "Dynamic edge fabric environment: Seamless and automatic switching among resources at the edge of IoT network and cloud," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2019, pp. 77–86.

[23] J. S. Ng *et al.*, "A survey of coded distributed computing," 2020. [Online]. Available: arXiv:2008.09048.

[24] H. Kuwahara, Y. Hsu, K. Matsuda, and M. Matsuoka, "Real-time workload allocation optimizer for computing systems by using deep learning," in *Proc. IEEE 12th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2019, pp. 190–192.

[25] H. Feng, J. Llorca, A. M. Tulino, and A. F. Molisch, "Dynamic network service optimization in distributed cloud networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2016, pp. 300–305.

[26] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "VNF placement and resource allocation for the support of vertical services in 5G networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 433–446, Feb. 2019.

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

[28] F. B. Jemaa, G. Pujolle, and M. Pariente, "QoS-aware VNF placement optimization in edge-central carrier cloud architecture," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–7.

[29] 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)*, Jul. 2016, pp. 291–300.

[30] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-aware resource allocation for edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jun. 2017, pp. 47–54.

[31] J. Sung, S. Han, and J. Kim, "Virtual machine pre-provisioning for computation offloading service in edge cloud," in *Proc. IEEE 12th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2019, pp. 490–492.

[32] J. Liu, K. Xiong, D. W. K. Ng, P. Fan, and Z. Zhong, "Optimal design of wireless-powered hierarchical fog-cloud computing networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[33] W. Du *et al.*, "Multiple energy harvesting devices enabled joint computation offloading and dynamic resource allocation for mobile-edge computing systems," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2019, pp. 154–158.

[34] F. Wang, H. Xing, and J. Xu, "Optimal resource allocation for wireless powered mobile edge computing with dynamic task arrivals," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[35] A. B. de Souza, P. A. L. Rego, and J. N. de Souza, "Exploring computation offloading in vehicular clouds," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–4.

[36] Q.-V. Pham *et al.*, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," 2020. [Online]. Available: arxiv.abs/1906.08452.

[37] 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.

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

[39] T. H. Noor, S. Zeadally, A. Alfazi, and Q. Z. Sheng, "Mobile cloud computing: Challenges and future research directions," *J. Netw. Comput. Appl.*, vol. 115, pp. 70–85, May 2018.

[40] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, Jun. 2013.

[41] B. Yi, X. Wang, K. Li, S. k. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Comput. Netw.*, vol. 133, pp. 212–262, Mar. 2018.

[42] J. Zhang, H. Huang, and X. Wang, "Resource provision algorithms in cloud computing," *J. Netw. Comput. Appl.*, vol. 64, pp. 23–42, Apr. 2016.

[43] 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.

[44] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 38–67, 1st Quart., 2020.

[45] J. Moura and D. Hutchison, "Game theory for multi-access edge computing: Survey, use cases, and future trends," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 260–288, 1st Quart., 2019.

[46] K. Toczé and S. Nadjm-Tehrani, "A taxonomy for management and optimization of multiple resources in edge computing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–23, Jun. 2018. [Online]. Available: https://www.hindawi.com/journals/wcmc/2018/7476201/

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

[48] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1409–1434, 2nd Quart., 2019.

[49] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5G network infrastructure," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 3879–3884.

[50] M. Mechtri, C. Ghribi, and D. Zeghlache, "VNF placement and chaining in distributed cloud," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2016, pp. 376–383.

[51] A. Zhou *et al.*, "Cloud service reliability enhancement via virtual machine placement optimization," *IEEE Trans. Services Comput.*, vol. 10, no. 6, pp. 902–913, Nov./Dec. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7387769/

[52] D. Gupta and J. Kuri, "Optimal VNF placement and traffic steering using column generation," in *Proc. IEEE 9th Int. Conf. Cloud Netw. (CloudNet)*, 2020, pp. 1–4.

[53] B. Németh, Y. Pignolet, M. Rost, S. Schmid, and B. Vass, "Cost-efficient embedding of virtual networks with and without routing flexibility," in *Proc. IFIP Netw. Conf. (Netw.)*, 2020, pp. 476–484.

[54] R. Mahmud, A. N. Toosi, K. Rao, and R. Buyya, "Context-aware placement of industry 4.0 applications in fog computing environments," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7004–7013, Nov. 2020.

[55] 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.

[56] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 3909–3914.

[57] R. Behravesh, E. Coronado, D. Harutyunyan, and R. Riggio, "Joint user association and VNF placement for latency sensitive applications in 5G networks," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–7.

[58] C. Morin, G. Texier, C. Caillouet, G. Desmangles, and C. Phan, "VNF placement algorithms to address the mono- and multi-tenant issues in edge and core networks," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–6.

[59] V. Farhadi *et al.*, "Service placement and request scheduling for data-intensive applications in edge clouds," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1279–1287.

[60] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[61] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*, May 2017, pp. 1222–1228.

[62] F. A. Salaht, F. Desprez, A. Lebre, C. Prud'homme, and M. Abderrahim, "Service placement in fog computing using constraint programming," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Jul. 2019, pp. 19–27.

[63] P. Lai *et al.*, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. Service Orient. Comput.*, 2018, pp. 230–245.

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

[65] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware VNF placement for service-customized 5G network slices," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 2449–2457.

[66] D. Haja, B. Vass, and L. Toka, "Improving big data application performance in edge-cloud systems," in *Proc. IEEE 12th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2019, pp. 187–189.

[67] D. Haja, B. Vass, and L. Toka, "Towards making big data applications network-aware in edge-cloud systems," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–6.

[68] F. Faticanti, F. De Pellegrini, D. Siracusa, D. Santoro, and S. Cretti, "Cutting throughput with the edge: App-aware placement in fog computing," in *Proc. 6th IEEE Int. Conf. Cyber Security Cloud Comput. (CSCloud) 5th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Jun. 2019, pp. 196–203.

[69] 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.

[70] C. Guerrero, I. Lera, and C. Juiz, "Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures," *Future Gener. Comput. Syst.*, vol. 97, pp. 131–144, Aug. 2019.

[71] V. Sundararaj, "Optimal task assignment in mobile cloud computing by queue based ant-bee algorithm," *Wireless Pers. Commun.*, vol. 104, no. 1, pp. 173–197, Sep. 2018.

[72] M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. V. Vasilakos, "On optimal and fair service allocation in mobile cloud computing," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 815–828, Jul. 2018.

[73] D. Happ, S. Bayhan, and V. Handziski, "JOI: Joint placement of IoT analytics operators and pub/sub message brokers in fog-centric IoT platforms," *Future Gener. Comput. Syst.*, vol. 119, pp. 7–19, Jun. 2021.

[74] 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.

[75] M. Zakarya *et al.*, "epcAware: A game-based, energy, performance and cost efficient resource management technique for multi-access edge computing," *IEEE Trans. Services Comput.*, early access, Jun. 26, 2020, doi: 10.1109/TSC.2020.3005347.

[76] R. Fantacci and B. Picano, "A matching game with discard policy for virtual machines placement in hybrid cloud-edge architecture for industrial IoT systems," *IEEE Trans. Ind. Informat.*, vol. 16, no. 11, pp. 7046–7055, Nov. 2020.

[77] M. Teng, X. Li, X. Qin, and J. Wu, "Priority based service placement strategy in heterogeneous mobile edge computing," in *Algorithms and Architectures for Parallel Processing*. Cham, Switzerland: Springer Int., 2020, pp. 314–329.

[78] P. Maiti, J. Shukla, B. Sahoo, and A. K. Turuk, "QoS-aware fog nodes placement," in *Proc. 4th Int. Conf. Recent Adv. Inf. Technol. (RAIT)*, Mar. 2018, pp. 1–6.

[79] A. Ceselli, M. Premoli, and S. Secci, "Cloudlet network design optimization," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, May 2015, pp. 1–9.

[80] H. Cheng, W. Xia, F. Yan, and L. Shen, "Balanced clustering and joint resources allocation in cooperative fog computing system," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[81] Q. Pham, T. Leanh, N. H. Tran, B. J. Park, and C. S. Hong, "Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach," *IEEE Access*, vol. 6, pp. 75868–75885, 2018.

[82] L. Zhao, J. Liu, Y. Shi, W. Sun, and H. Guo, "Optimal placement of virtual machines in mobile edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[83] K. Li and J. Nabrzyski, "Traffic-aware virtual machine placement in cloudlet mesh with adaptive bandwidth," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2017, pp. 49–56.

[84] W. Ma, X. Liu, and L. Mashayekhy, "A strategic game for task offloading among capacitated UAV-mounted cloudlets," in *Proc. IEEE Int. Congr. Internet Things (ICIOT)*, Jul. 2019, pp. 61–68.

[85] S. Zhu, L. Gui, J. Chen, Q. Zhang, and N. Zhang, "Cooperative computation offloading for UAVs: A joint radio and computing resource allocation approach," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 74–79.

[86] Y. Song, S. S. Yau, R. Yu, X. Zhang, and G. Xue, "An approach to QoS-based task distribution in edge computing networks for IoT applications," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jun. 2017, pp. 32–39.

[87] 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.

[88] Z. Xu, W. Gong, Q. Xia, W. Liang, O. Rana, and G. Wu, "NFV-enabled IoT service provisioning in mobile edge clouds," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 1892–1906, May 2021.

[89] P. Kayal and J. Liebeherr, "Distributed service placement in fog computing: An iterative combinatorial auction approach," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 2145–2156.

[90] M. Huang, W. Liang, X. Shen, Y. Ma, and H. Kan, "Reliability-aware virtualized network function services provisioning in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2699–2713, Nov. 2020.

[91] R. Gouareb, V. Friderikos, and A. Aghvami, "Virtual network functions routing and placement for edge cloud latency minimization," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2346–2357, Oct. 2018.

[92] Z. Chen *et al.*, "A novel algorithm for NFV chain placement in edge computing environments," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[93] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 108–119, Jan.–Mar. 2017.

[94] M. S. Elbamby, M. Bennis, W. Saad, M. Latva-Aho, and C. S. Hong, "Proactive edge computing in fog networks with latency and reliability guarantees," *EURASIP J. Wireless Commun. Netw.*, vol. 2018, no. 1, p. 209, 2018. [Online]. Available: https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-018-1218-y#citeas

[95] H. D. Chantre and N. L. S. da Fonseca, "Multi-objective optimization for edge device placement and reliable broadcasting in 5G NFV-based small cell networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2304–2317, Oct. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8463561/

[96] Y. Yue, B. Cheng, B. Li, M. Wang, and X. Liu, "Throughput optimization VNF placement for mapping SFC requests in MEC-NFV enabled networks," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2020, pp. 1–3.

[97] M. Chen, Y. Sun, H. Hu, L. Tang, and B. Fan, "Energy-saving and resource-efficient algorithm for virtual network function placement with network scaling," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 1, pp. 29–40, Mar. 2021.

[98] 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.

[99] A. Brogi, S. Forti, C. Guerrero, and I. Lera, "Meet genetic algorithms in Monte Carlo: Optimised placement of multi-service applications in the fog," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2019, pp. 13–17.

[100] 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.

[101] N. Eshraghi and B. Liang, "Joint offloading decision and resource allocation with uncertain task computing requirement," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1414–1422.

[102] T. T. Vu, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "QoS-aware fog computing resource allocation using feasibility-finding benders decomposition," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[103] Z. Jiang and S. Mao, "Energy delay tradeoff in cloud offloading for multi-core mobile devices," *IEEE Access*, vol. 3, pp. 2306–2316, 2015.

[104] V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, and G. Tashakor, "Handling service allocation in combined Fog-cloud scenarios," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–5.

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

[106] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, and R. H. Glitho, "Application component placement in NFV-based hybrid cloud/fog systems with mobile fog nodes," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1130–1143, May 2019.

[107] H. Jin, X. Zhu, and C. Zhao, "Computation offloading optimization based on probabilistic SFC for mobile online gaming in heterogeneous network," *IEEE Access*, vol. 7, pp. 52168–52180, 2019.

[108] E. E. Haber, H. Alameddine, and C. Assi, "A reliability-aware computation offloading solution via UAV-mounted cloudlets," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–6.

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

[110] H. Zhao, S. Deng, C. Zhang, W. Du, Q. He, and J. Yin, "A mobility-aware cross-edge computation offloading framework for partitionable applications," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2019, pp. 193–200.

[111] G. Zheng, A. Tsiopoulos, and V. Friderikos, "Dynamic VNF chains placement for mobile IoT applications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[112] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[113] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[114] S. Pasteris, S. Wang, M. Herbster, and T. He, "Service placement with provable guarantees in heterogeneous edge computing systems," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 514–522.

[115] 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.

[116] P.-D. Nguyen and L. B. Le, "Joint computation offloading, SFC placement, and resource allocation for multi-site MEC systems," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2020, pp. 1–6.

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

[118] J. Martín-Pérez and C. J. Bernardos, "Multi-domain VNF mapping algorithms," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2018, pp. 1–6.

[119] R. G. Aryal and J. Altmann, "Dynamic application deployment in federations of clouds and edge resources using a multiobjective optimization AI algorithm," in *Proc. 3rd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Apr. 2018, pp. 147–154.

[120] A. Mohamad and H. S. Hassanein, "On demonstrating the gain of SFC placement with VNF sharing at the edge," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[121] F. Hao, M. Kodialam, T. V. Lakshman, and S. Mukherjee, "Online allocation of virtual machines in a distributed cloud," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 238–249, Feb. 2017.

[122] S. Wang, M. Zafer, and K. K. Leung, "Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017.

[123] B. Németh, B. Sonkoly, M. Rost, and S. Schmid, "Efficient service graph embedding: A practical approach," in *Proc. IEEE Conf. Netw. Function Virtual. Softw. Defined Netw. (O4SDI @ IEEE NFV-SDN)*, Nov. 2016, pp. 19–25.

[124] B. Sonkoly *et al.*, "5G applications from vision to reality: Multi-operator orchestration," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 7, pp. 1401–1416, Jul. 2020.

[125] W. Zhang, S. Li, L. Liu, Z. Jia, Y. Zhang, and D. Raychaudhuri, "Hetero-edge: Orchestration of real-time vision applications on heterogeneous edge clouds," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1270–1278.

[126] M. Wang, B. Cheng, W. Feng, and J. Chen, "An efficient service function chain placement algorithm in a MEC-NFV environment," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[127] 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.

[128] M. Wang, B. Cheng, and J. Chen, "An efficient service function chaining placement algorithm in mobile edge computing," *ACM Trans. Internet Technol.*, vol. 20, no. 4, pp. 1–21, Nov. 2020. [Online]. Available: https://doi.org/10.1145/3388241

[129] G. Castellano, F. Esposito, and F. Risso, "A distributed orchestration algorithm for edge computing resources with guarantees," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 2548–2556.

[130] Q. Peng *et al.*, "Mobility-aware and migration-enabled online edge user allocation in mobile edge computing," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2019, pp. 91–98.

[131] T. Bahreini and D. Grosu, "Efficient placement of multi-component applications in edge computing systems," in *Proc. 2nd ACM/IEEE Symp. Edge Comput. (SEC)*, Oct. 2017, pp. 1–11.

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

[133] Y. Yu, T. Chiu, A. Pang, M. 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.

[134] H. Zhu and C. Huang, "Availability-aware mobile edge application placement in 5G networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[135] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards edge slicing: VNF placement algorithms for a dynamic amp; realistic edge cloud environment," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2017, pp. 1–6.

[136] T. Wang, X. Wei, T. Liang, and J. Fan, "Dynamic tasks scheduling based on weighted bi-graph in mobile cloud computing," *Sustain. Comput. Informat. Syst.*, vol. 19, pp. 214–222, May 2018.

[137] N. Kiran, X. Liu, S. Wang, and C. Yin, "VNF placement and resource allocation in SDN/NFV-enabled MEC networks," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, 2020, pp. 1–6.

[138] Y. Liu, G. Feng, G. Zhao, Z. Chen, and S. Qin, "Virtual network function deployment strategy in clustered multi-mobile edge clouds," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2020, pp. 1–6.

[139] B. Liu, X. Xu, L. Qi, Q. Ni, and W. Dou, "Task scheduling with precedence and placement constraints for resource utilization improvement in multi-user MEC environment," *J. Syst. Architect.*, vol. 114, Mar. 2021, Art. no. 101970.

[140] T. Bahreini and D. Grosu, "Efficient algorithms for multi-component application placement in mobile edge computing," *IEEE Trans. Cloud Comput.*, early access, Nov. 17, 2020, doi: 10.1109/TCC.2020.3038626.

[141] D. Y. Zhang and D. Wang, "An integrated top-down and bottom-up task allocation approach in social sensing based edge computing systems," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 766–774.

[142] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[143] S. Josilo and G. Dán, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 2467–2475.

[144] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[145] X. Wu, J. Duan, M. Zhong, P. Li, and J. Wang, "VNF chain placement for large scale IoT of intelligent transportation," *Sensors*, vol. 20, no. 14, p. 3819, Jul. 2020.

[146] B. Nemeth, N. Molner, J. Martinperez, C. J. Bernardos, A. De la Oliva, and B. Sonkoly, "Delay and reliability-constrained VNF placement on mobile and volatile 5G infrastructure," *IEEE Trans. Mobile Comput.*, early access, Jan. 28, 2021, doi: 10.1109/TMC.2021.3055426.

[147] L. Zhang, J. Wu, S. Mumtaz, J. Li, H. Gacanin, and J. J. P. C. Rodrigues, "Edge-to-edge cooperative artificial intelligence in smart cities with on-demand learning offloading," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[148] 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.

[149] M. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 6790–6805, Oct. 2018.

[150] M. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2868–2881, Dec. 2018.

[151] C. Avasalcai, C. Tsigkanos, and S. Dustdar, "Decentralized resource auctioning for latency-sensitive edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2019, pp. 72–76.

[152] H. Zhang, Y. Xiao, S. Bu, D. Niyato, R. Yu, and Z. Han, "Fog computing in multi-tier data center networks: A hierarchical game approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[153] A. Mehta and E. Elmroth, "Distributed cost-optimized placement for latency-critical applications in heterogeneous environments," in *Proc. IEEE Int. Conf. Auton. Comput. (ICAC)*, Sep. 2018, pp. 121–130.

[154] Q. Pham, L. B. Le, S. Chung, and W. Hwang, "Mobile edge computing with wireless backhaul: Joint task offloading and resource allocation," *IEEE Access*, vol. 7, pp. 16444–16459, 2019.

[155] W. Du *et al.*, "Service capacity enhanced task offloading and resource allocation in multi-server edge computing environment," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2019, pp. 83–90.

[156] M. A. Hassan, M. Xiao, Q. Wei, and S. Chen, "Help your mobile applications with fog computing," in *Proc. 12th Annu. IEEE Int. Conf. Sens. Commun. Netw. Workshops (SECON Workshops)*, Jun. 2015, pp. 1–6.

[157] I. Alghamdi, C. Anagnostopoulos, and D. P. Pezaros, "Delay-tolerant sequential decision making for task offloading in mobile edge computing environments," *MDPI Inf.*, vol. 10, no. 10, p. 312, Oct. 2019.

[158] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Video transcoding, caching, and multicast for heterogeneous networks over wireless network virtualization," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 141–144, Jan. 2018.

[159] 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.

[160] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power IoT edge devices," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 7–12.

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

[162] D. Haja *et al.*, "How to orchestrate a distributed OpenStack," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 293–298.

[163] M. Szabo, D. Haja, and M. Szalay, "Cost-efficient resource allocation method for heterogeneous cloud environments," *HTE Infocommun. J.*, vol. 10, no. 1, pp. 15–21, 2018.

[164] M. Szalay, D. Haja, J. Doka, B. Sonkoly, and L. Toka, "Demo abstract: Turning openstack into a fog orchestrator," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 947–948.

[165] 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.

[166] L. Toka, D. Haja, A. Korosi, and B. Sonkoly, "Resource provisioning for highly reliable and ultra-responsive edge applications," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–6.

[167] B. Sonkoly *et al.*, "Scalable edge cloud platforms for IoT services," *J. Netw. Comput. Appl.*, vol. 170, Nov. 2020, Art. no. 102785. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804520302599

[168] A. Mseddi, W. Jaafar, H. Elbiaze, and W. Ajib, "Intelligent resource allocation in dynamic fog computing environments," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–7.

[169] 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.

[170] W. Tärneberg *et al.*, "Dynamic application placement in the mobile cloud network," *Future Gener. Comput. Syst.*, vol. 70, pp. 163–177, May 2017.

[171] B. Németh *et al.*, "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.

[172] M. Golkarifard, C. F. Chiasserini, F. Malandrino, and A. Movaghar, "Dynamic VNF placement, resource allocation and traffic routing in 5G," *Comput. Netw.*, vol. 188, Jan. 2021, Art. no. 107830.

[173] H. Xiang *et al.*, "An adaptive cloudlet placement method for mobile applications over GPS big data," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[174] A. Ceselli, M. Premoli, and S. Secci, "Mobile edge cloud network design optimization," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1818–1831, Jun. 2017.

[175] 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.

[176] 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.

[177] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Profit-aware application placement for integrated fog-cloud computing environments," *J. Parallel Distrib. Comput.*, vol. 135, pp. 177–190, Jan. 2020.

[178] 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.

[179] Y. Chiang, Y.-H. Chao, C.-H. Hsu, C.-T. Chou, and H.-Y. Wei, "Virtual network embedding with dynamic speed switching orchestration in fog/edge network," *IEEE Access*, vol. 8, pp. 84753–84768, 2020.

[180] 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.

[181] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Energy-aware application placement in mobile edge computing: A stochastic optimization approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 909–922, Apr. 2020.

[182] P. Roy *et al.*, "AI-enabled mobile multimedia service instance placement scheme in mobile edge computing," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107573.

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

[184] D. Haja, M. Szalay, B. Sonkoly, G. Pongracz, and L. Toka, "Sharpening kubernetes for the edge," in *Proc. ACM SIGCOMM Conf. Posters Demos*, Aug. 2019, pp. 136–137.

[185] M. Rost and S. Schmid, "Charting the complexity landscape of virtual network embeddings," in *Proc. IFIP Netw. Conf. (IFIP Netw.) Workshops*, May 2018, pp. 1–9.

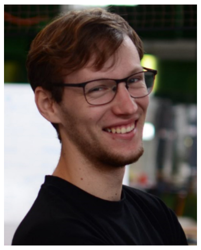
[186] *FogAtlas*. Accessed: Oct. 1, 2020. [Online]. Available: https://fogatlas.fbk.eu/

[187] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable network design library," in *Proc. 3rd Int. Netw. Optim. Conf. (INOC)*, 2007, p. 9.
[188] J. Chan and A. Seneviratne, "A practical user mobility prediction algorithm for supporting adaptive QoS in wireless networks," in *Proc. IEEE Int. Conf. Netw. (ICON)*, 1999, pp. 104–111.
[189] M. Manalastas, H. Farooq, S. M. A. Zaidi, and A. Imran, "Where to go next? A realistic evaluation of Ai-assisted mobility predictors for HetNets," in *Proc. IEEE 17th Annu. Consum. Commun. Netw. Conf. (CCNC)*, 2020, pp. 1–6.
[190] D. Haja, Z. R. Turanyi, and L. Toka, "Location, proximity, affinity—The key factors in FaaS," *Infocommun. J.*, vol. 12, no. 4, pp. 14–21, 2020.
[191] M. Szalay, P. Mátray, and L. Toka, "State management for cloud-native applications," *Electronics*, vol. 10, no. 4, p. 423, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/4/423
[192] M. Szalay, P. Mátray, and L. Toka, "Minimizing state access delay for cloud-native network functions," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, 2019, pp. 1–6.
[193] M. Kiamari, C. Wang, and A. S. Avestimehr, "On heterogeneous coded distributed computing," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, 2017, pp. 1–7.
[194] N. Woolsey, R.-R. Chen, and M. Ji, "Coded distributed computing with heterogeneous function assignments," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6.
[195] *Horizontal Pod Autoscaler in Kubernetes*. Accessed: Oct. 1, 2020. [Online]. Available: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/
[196] L. Toka, G. Dobreff, B. Fodor, and B. Sonkoly, "Machine learning-based scaling management for kubernetes edge clusters," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 958–972, Mar. 2021.
[197] 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.
[198] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18209–18237, 2018.
[199] X. Xu, X. Liu, Z. Xu, F. Dai, X. Zhang, and L. Qi, "Trust-oriented IoT service placement for smart cities in edge computing," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4084–4091, May 2020.
[200] R. Lu, K. Heung, A. H. Lashkari, and A. A. Ghorbani, "A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT," *IEEE Access*, vol. 5, pp. 3302–3312, 2017.
[201] C. Xu, J. Ren, D. Zhang, and Y. Zhang, "Distilling at the edge: A local differential privacy obfuscation framework for IoT data analytics," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 20–25, Aug. 2018.
[202] Z. Zhao, W. Zhou, D. Deng, J. Xia, and L. Fan, "Intelligent mobile edge computing with pricing in Internet of Things," *IEEE Access*, vol. 8, pp. 37727–37735, 2020.
[203] I. Pelle, J. Czentye, J. Dóka, A. Kern, B. P. Geró, and B. Sonkoly, "Operating latency sensitive applications on public serverless edge cloud platforms," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 7954–7972, May 2021.
[204] T. Elgamal, "Costless: Optimizing cost of serverless computing through function fusion and placement," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, 2018, pp. 300–312.

Balázs Sonkoly received the M.Sc. and Ph.D. degrees in computer science from Budapest University of Technology and Economics (BME) in 2002 and 2010, respectively. He is an Associate Professor with the BME, where he is the Head of MTA-BME Network Softwarization Research Group. He has participated in several EU projects (FP7 OpenLab, FP7 UNIFY, and H2020 5G Exchange) and national projects. His current research activity focuses on cloud/edge/fog computing, NFV, SDN, and 5G. He was the Demo Co-Chair of ACM SIGCOMM 2018, EWSDN'15,'14, and IEEE HPSR'15.

János Czentye received the M.Sc. degree (Highest Hons.) in the topic of networks and services in 2014. He is currently pursuing the Ph.D. degree with the Budapest University of Technology and Economics. He worked with the HSNLab, participating in European research projects FP7 UNIFY and H2020 5GEx and gained wide knowledge about SDN/NFV, microservices, and cloud technologies, for four years. His current Ph.D. research focuses on cloud-native service modeling, composition, and provisioning.

Márk Szalay is currently pursuing the Ph.D. degree with HSNLab, Budapest University of Technology and Economics. His main research interests include hardware (router/switch/NIC) design, network programming, software-defined networking, and network function virtualization.



Balázs Németh received the M.Sc. degree from the Budapest University of Technology and Economics (BME), as a Computer Science Engineer in infocommunication specialization in 2016. He is currently pursuing the Ph.D. degree in network softwarization with special focus on orchestration algorithms and next generation network models. He is an Industrial Ph.D. student with BME, in cooperation with Ericsson Research. He has been working on orchestration algorithms for the H2020 5G-PPP 5G Exchange project.

László Toka (Member, IEEE) received the Ph.D. degree from Telecom ParisTech in 2011. He is an Assistant Professor with the Budapest University of Technology and Economics, where he is the Vice-Head of HSNLab and a Member of the MTA-BME Network Softwarization and the MTA-BME Information Systems Research Groups. He worked with Ericsson Research from 2011 and 2014, then he joined the academia with research focus on software-defined networking, cloud computing, and artificial intelligence.