Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey

THANG LE DUC, Umeå University, Sweden
RAFAEL GARCÍA LEIVA, IMDEA Networks Institute, Spain
PAOLO CASARI, IMDEA Networks Institute, Spain
PER-OLOV ÖSTBERG, Umeå University, Sweden

Large-scale software systems are currently designed as distributed entities and deployed in cloud data centers. To overcome the limitations inherent to this type of deployments, applications are increasingly being supplemented with components instantiated closer to the edges of networks – a paradigm known as edge computing. The problem of how to efficiently orchestrate combined edge-cloud applications is however incompletely understood, and a wide range of techniques for resource and application management are currently in use.

This paper investigates the problem of reliable resource provisioning in joint edge-cloud environments and surveys technologies, mechanisms, and methods that can be used to improve the reliability of distributed applications in diverse and heterogeneous network environments. Due to the complexity of the problem, special emphasis is placed on solutions to the characterization, management, and control of complex distributed applications using machine learning approaches. The survey is structured around a decomposition of the reliable resource provisioning problem into three categories of techniques: workload characterization and prediction, component placement and system consolidation, and application elasticity and remediation. Survey results are presented along with a problem-oriented discussion of the state of the art. Finally, a summary of identified challenges and an outline of future research directions are presented to conclude the paper.

CCS Concepts: • **Computer systems organization** → **Cloud computing**; **Reliability**; **Availability**; • **Computing methodologies** → **Model development and analysis**;

Additional Key Words and Phrases: reliability; cloud computing; edge computing; distributed systems; placement; consolidation; autoscaling; remediation; machine learning; optimization

ACM Reference Format:
Thang Le Duc, Rafael García Leiva, Paolo Casari, and Per-Olov Östberg. 2019. Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey. *ACM Comput. Surv.* 00, 0, Article 1 (June 2019), 35 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

The past decades have witnessed the trend of moving computation, data storage and applications into cloud data centers (DCs), facilitating ubiquitous access to shared computing and storage resources on demand. The cloud computing model demonstrates numerous advantages, including high utilization of shared resources, low cost in service deployment and management, high scalability, accessibility, and availability [129]. Today, in the era of the Internet of Things (IoT), an unprecedented volume

Authors' addresses: Thang Le Duc, Umeå University, Umeå, 90187, Sweden, thang@cs.umu.se; Rafael García Leiva, IMDEA Networks Institute, Madrid, 28918, Spain, rafael.garcia@imdea.org; Paolo Casari, IMDEA Networks Institute, Madrid, 28918, Spain, paolo.casari@imdea.org; Per-Olov Östberg, Umeå University, Umeå, 90187, Sweden, p-o@cs.umu.se.

0360-0300/2019/6-ART1 $15.00
https://doi.org/0000001.0000001

of data generated by the IoT devices needs to be collected and analyzed [17]. The collection of such volumes of data in cloud DCs incurs extremely high latency and network bandwidth usage. To address this issue, the *fog* or *edge computing* paradigm has been proposed, which pushes the data analysis closer to the systems or devices that data originates from [17, 75]. It is however unclear how to orchestrate complex, large-scale software systems in heterogeneous network environments, and the topic of such orchestration, or how to automate computing platforms that integrate both edge and cloud computing capabilities, is the subject of significant research efforts. Fig. 1 illustrates hierarchical tiers of an (heterogeneous) edge-cloud network.

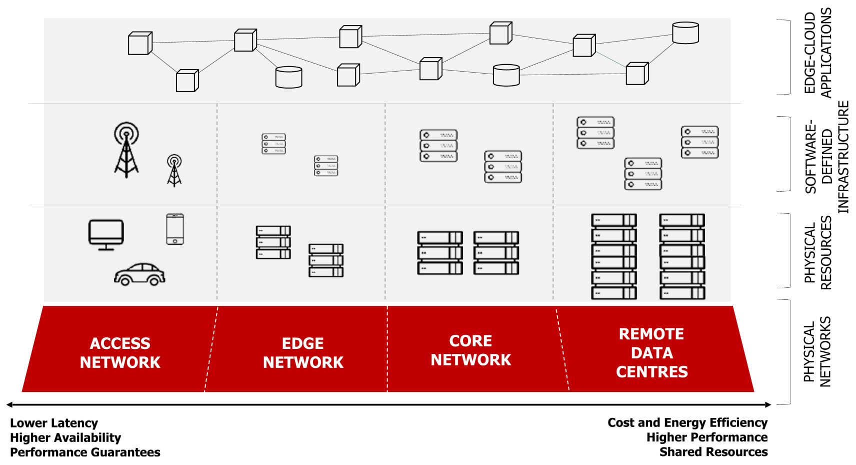

Fig. 1. A conceptual illustration of edge computing. To improve the application performance and the network bandwidth utilization, selected components are deployed closer to end-users in core and edge networks.

This paper targets techniques for reliable resource provisioning that can operate across different (and often multiple) distributed computing system domains, including cloud, edge and fog computing. In this respect, we do not limit our scope to a specific scenario, context or ecosystem, but rather focus on the techniques proposed to tackle provisioning, elasticity and remediation problems, and discuss how these techniques are shaping the next generation of edge/fog/cloud computing architectures. Such architectures are extremely heterogeneous and complex in many respects, to the point that a "fully engineered" approach to modeling and prediction/actuation may be exceedingly complex, if not plainly ineffective. For this reason, we explicitly target those techniques that employ some form of machine learning applied to modeling, prediction, classification and forecasting related to resource provisioning, applications/components placement, migration, and remediation in a continuum of distributed computing contexts. Accordingly, this paper first decomposes and structures the problem of reliable resource provisioning, and then puts a special emphasis on a survey of machine learning-based approaches to the solution of this problem or parts thereof.

Several recent surveys have addressed different aspects of cloud, edge and fog computing to date. Among these, [85] surveys the current status of the integration of cloud computing and edge computing for the IoT. The advantages and weaknesses of several available architectures and solutions are discussed along with comments about the IoT applications that can benefit from distributed cloud deployments. The work in [105] surveys 105 works about mechanisms for the identification and optimal provision of cloud resources, chosen based on their quality and relevance. The authors propose a collection of research questions, match the papers with these questions, and identify covered and less investigated areas. Moreover, they propose future directions, such as studying the impact of Service Level Agreement (SLA) violations, dealing with different Quality of Service (QoS) parameters, and a need for evaluations of the proposed solutions in real cloud environments. Numerous architectures of edge-oriented computing systems are explored in [58], which proposes to clarify the edge/fog terminology, through related concepts. The main focus of the survey is on different architectures, management and optimization approaches tackling a number of issues in the architectures, including resource provisioning, placement, offloading, migration. A description of prior work based on pursued optimization objectives is also provided.

While very useful to put distributed edge/fog/cloud computing systems in perspective and to assess the progress of the recent research and engineering of these systems, none of the above surveys provides an analysis of the available literature from a machine learning perspective. Among other propositions, our paper aims at filling this gap (which has been only partially addressed by such works as [113]), and especially focuses on machine learning-based techniques, thus highlighting the opportunities that these techniques provide for the management of complex distributed systems. In more detail, the contributions of this paper include: (1) a problem formulation of reliable resource provisioning for distributed applications in edge-cloud environments; (2) a decomposition of the problem and a classification of sub-problems regarding resource provisioning and application remediation; (3) a comprehensive review of cutting-edge classic and machine learning-based schemes for workload modeling, load balancing, application placement and migration, computing offloading, system autoscaling and remediation; and (4) a problem-oriented discussion of a next generation of edge-cloud computing techniques needed for addressing the problem.

The remainder of this paper is organized as follows. Section 2 introduces terminology and describes enabling technologies to support the survey. Section 3 defines the problem of reliable resource provisioning in edge-cloud environments, and provides a classification of sub-problems that form the structure of the survey. Section 4 surveys classic and machine learning-based methods for addressing various aspects of resource provisioning and compares the schemes. Section 5 presents a problem-oriented discussion about the state of the art, gives an overview of the surveyed solutions, identifies open issues and challenges, and discusses an optimization framework to fully address the problem. Finally, we draw concluding remarks in Section 6.

2 BACKGROUND, TERMINOLOGY, AND ENABLING TECHNOLOGY

2.1 Cloud Computing

Cloud computing concerns the provisioning of resources, including computation, memory, storage, network, and applications/services, over the Internet. This computing paradigm basically adopts the client-server architecture and facilitates centralized deployment and computation offloading for applications. In this way, cloud computing is cost-efficient in application deployment and maintenance, and flexible in resource provisioning and in decoupling services from underlying technologies at both the client and server side. Cloud computing and its enabling technologies have been studied for decades, and numerous mature computing platforms have been delivered in the market, e.g., Amazon EC2, Google Cloud Platform, Microsoft Azure and IBM SmartCloud.

The expeditious evolution of mobility-enabling technologies along with the popularity of mobile devices nowadays has pushed the research on cloud computing to support mobile applications as well as user/device mobility. Modern mobile devices are equipped with powerful sensing capabilities, hence, can provide sensory data of their surrounding areas. By exploiting such data, the devices and applications are able to bring context-aware services to users. Because of this trend, *mobile cloud computing* has been introduced in [49] as an integration of mobile computing and cloud computing. It is formally defined as a novel computing paradigm aiming at resource provisioning to the devices so as to boost the context awareness capability of both the devices and applications. The work also surveys numerous platforms that have been developed for this computing paradigm.

2.2 Edge and Fog Computing

A huge volume of data needs to be aggregated and consumed in order to generate intelligence and improve the context awareness of distributed applications. Fog and edge computing can meet this demand by leveraging distributed resources, up to the edge of a network. In the fog computing paradigm [78], the continuum of resources from the cloud to the edge is leveraged to provision computing and storage capacity to services and applications. Edge computing extends the concept further, so that potentially all computation and storage resources of edge network elements, e.g., access points, switches, routers, base stations and mobile devices, are utilized to provide a cloud-like computing environment to applications [94]. Besides an extension in the utilization of network infrastructure, this also provides optimizations in replicating information across networks to enhance local context awareness. Considering the convergence of information technology (IT) and telecommunications networks, *mobile edge computing* (MEC) is proposed as an instance of edge computing, in which the computational capabilities of Cloud and IT are integrated in Radio Access Networks (RANs) which are close to subscribers [44, 101]. For standardization purposes, the European Telecommunications Standards Institute (ETSI) has established an industry standard for MEC as depicted in [41], which targets at an incorporation of the telco and IT cloud systems to form cloud-like environments within RANs.

Cloudlet computing is another architecture under the umbrella of edge computing [98]. A cloudlet is known as a small-scale DC bridging mobile/IoT devices and the central cloud. It is dedicated to quickly provision cloud services to the devices within its vicinity [99]. To support cloudlet deployment, OpenStack++ [114] (an extension of OpenStack [82]) has been developed by Carnegie Mellon University to provide a set of cloudlet-specific APIs. Moreover, commercial cloudlet-based applications have recently appeared in the market, e.g. Akamai's cloudlet applications [47].

2.3 Application Architecture

The emergence of data-intensive and highly interconnected distributed applications such as fifth-generation (5G) mobile telecommunication systems, IoT applications, and big data frameworks are driving the need for new paradigms for the design and deployment of large-scale software systems. Under the cloud computing umbrella, various conceptual models have been introduced for distributed applications [118, 121]. A distributed application in the cloud generally consists of a set of components spread over multiple cloud resources, while in edge computing the components of an application can be deployed on nodes both inside and outside the central cloud (*edge-cloud application*) . The shift of components from the cloud towards the network's edge makes it possible for the application to lower its network footprint while improving its responsiveness. The aim of edge deployments is apparently to combine the strengths of edge environments (e.g., low latency, high localization, network offloading) and cloud DCs (e.g., scalability, energy efficiency, economy of scale effects) to construct systems that better meet the needs of data-intensive and/or latency-critical applications, especially when these needs are not cost-efficiently served by either approach alone.

Modern large-scale edge-cloud applications require novel architectures (beyond the notable multi-tier and service-oriented architectures) to flexibly adapt to changes in the underlying network infrastructures and computing techniques. One of the advanced architectures which has been introduced recently is the microservice architecture, simply referred to as *microservices*. According to this architecture, an application is implemented as a collection of loosely coupled services, each of which has it own business goal, communicates with others through a well-defined interface, mostly using HTTP/REST [27, 92], and can be developed, deployed and maintained independently. Microservices can be considered as an evolution of service-oriented architecture (SOA) to accommodate cloud and edge computing [104]. By employing microservices, a single cloud application can be distributedly deployed from cloud DCs down to the edges of the networks, and some components can be located even on the user devices. This architecture apparently delivers a high horizontal scaling flexibility in edge-cloud computing [68, 104]. As pointed out in [27, 68], Google, Netflix, Amazon, eBay, Twitter, and many other companies have evolved their applications toward microservices due to aforementioned benefits. Fig. 2 illustrates the architecture of a microservice-based application.

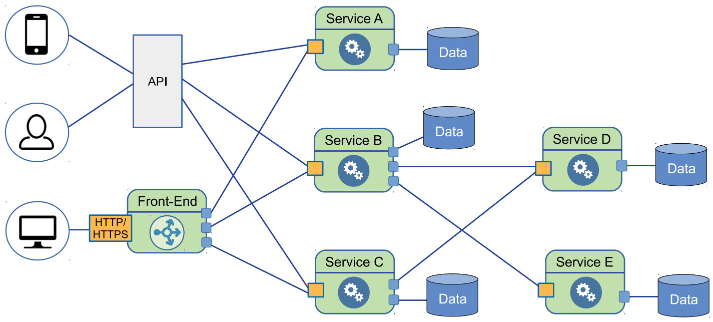

Fig. 2. An example of microservice-based application including one front-end service and five back-end services. The application can be accessed through a web (HTTP/HTTPS) interface or an API gateway.

2.4 Application Migration

Under the context of distributed edge-cloud application, application migration is defined as a relocation of one or multiple components within hosts or edge-/cloud-DCs. Since components are hosted and running in virtual machines (VMs) or containers, a component migration implies a VM/container migration. The execution of a migrated component needs to be resumed in the new host, thus the component's data related to the current VM/container instance's configuration, the component's state and user sessions have to be transferred to the host for initializing the environment and resuming the component. Migration aims to provide workload balancing, fault tolerance, system stability, and energy efficiency. Based on relocation strategies, it can be categorized as either live migration with almost zero service downtime or non-live migration with some service suspension [66, 81]. Non-live migration is the simplest and most naive technique [8], which first shuts down the source instance, transfers data of the instance image to the new host, and then reboot the instance. In contrast, live migration is more advanced and constitutes an attractive topic in both academia and industry. Numerous live migration schemes have been proposed [20, 108, 128], and various products currently support live migration [18], e.g. VMware ESX [116], Microsoft Hyper-V [71], Oracle VM Server [83], KVM [55], Google Cloud Platform.

2.5 Machine Learning

Machine learning is the discipline of teaching computers to predict outcomes or classify objects without being explicitly programmed for such tasks. One of its basic assumptions is that it is possible to build algorithms that are able to predict future, previously unseen values using training data and the application of statistical techniques. Machine learning has been highly successful in areas like self-driving cars, speech recognition, effective web search, and purchase recommendations [38], to name only a few examples. This success is mostly due to the availability of large datasets and the continuous improvements in the computational power of servers and GPUs [31].

Machine learning algorithms can be classified into two main groups: supervised and non-supervised algorithms. Supervised learning refers to building models given a collection of training predictors $X_1, X_2, \ldots, X_p$ and the corresponding response variable Y, whereas in unsupervised learning there exist only predictors, hence the algorithms have to learn the structure of the training data (clustering). When the goal is to predict a continuous or quantitative output value, the corresponding problem to be solved is called regression, whereas the prediction of a categorical or qualitative output is known as a classification problem. In Fig. 3 we provide a taxonomy of some of the most popular machine learning algorithms used in practice.

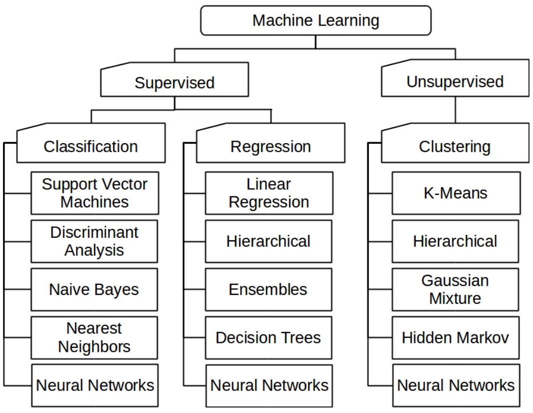

Fig. 3. Classification of the most common machine learning algorithms.

Machine learning methods can be parametric where certain assumptions are made about the functional form of the model and training data is then used to fit its parameters, e.g., as in polynomial regression, or non-parametric, e.g., neural networks. Machine learning can be used also for inference tasks, i.e., in order to understand how the response variable is affected when the predictors change.

3 RELIABLE RESOURCE PROVISIONING IN EDGE-CLOUD ENVIRONMENTS

As outlined in [84], reliable resource provisioning for edge-cloud applications is a complex problem, especially when it is examined in a multi-tenant edge-cloud environment where the infrastructure is utilized to host numerous applications/services owned by different service providers. Each application typically has its own set of requirements, and there is a high possibility that controlling operations or tuning the performance of one application will have some impact on the others. When dealing with user workloads, data analysis must be performed from two different angles, at both the application- and the infrastructure-level. Additionally, the problem also defines that the applications are deployed and operating in heterogeneous and geographically dispersed resource environments. In short, solving the problem completely and optimally is a challenging task which entails a mixture of multiple techniques to attain the predictability and controllability of the applications' performance in edge-cloud environments. This leads to the necessity of a comprehensive investigation of different aspects of the problem, so as to produce a holistic view on the challenges and issues when deriving a solution. For this purpose, in this section we outline and discuss aspects of resource management at the workload, application, and infrastructure levels.

3.1 Workload Analysis and Modeling

Workload is commonly interpreted, within the context of distributed application, as the total number of requests issued by clients to an application or a single application component. According to distinct characteristics and based on different perspectives, it can be categorized into different types, including sequential or non-sequential/random workload, transactional or non-transactional workload, and computation-intensive, data-intensive, or memory-sensitive workload [12, 33]. The demand of resources is different for each type of workload, in terms of both resource type and volume. The components of an edge-cloud application are designated to perform different tasks which highly likely deal with different kinds of workload. Moreover, because of the wide dispersion of the components over networks, workloads are distributed differently across locations from the cloud- to edge-DCs. These characteristics lead to a high complexity and challenges in workload analysis and modeling. Nevertheless, understanding the workload behavior is apparently beneficial for the performance and reliability improvement of an application, in the way that resources are sufficiently allocated for the application to serve any coming workloads. As a result, significant effort has been made to accomplish workload modeling recently, especially the work aiming at real workloads of Google [60, 61], Facebook [122], Netflix [107], and Wikimedia [9].

Furthermore, most recently comes a demand of workload behavior prediction [54, 61]. In the age of IoT, there emerges an increase of the number of mobile users/devices accessing cloud applications. This introduces sudden changes of workloads at certain locations within some periods of time. In other words, edge-cloud applications may experience variations of workloads in an unpredictable manner. Knowing about the spatio-temporal distribution of future workloads in advance brings benefits for application scaling, i.e., the system becomes capable of proactively scaling for an early resolution to meet the real-time resource needs of applications and services. In this way, resource utilization can be optimized while guaranteeing the SLA as well as the QoS, which in turn leverages the business of cloud service providers. For research and development, workload modeling and prediction enables the generation of proper models for constructing autonomous systems which are able to perform optimization in autoscaling and remediation (being discussed in the next sub-section) under diverse circumstances.

3.2 Application Optimization

Due to the nature of the computing platforms, the applications deployed in edge-cloud computing environments have to face numerous issues caused by the limited of bandwidth, the unreliability and heterogeneity of wireless communications, computation offloading, and security [24, 49]. The performance of such edge-cloud applications is also affected by the fluctuation of the workload of different components at different locations in the network. In order to maintain QoS, elasticity and remediation should be taken into account as two key requirements of an application [77]. In fact, elasticity enables computational resources to be flexibly scaled to meet the demand of applications or the users, while remediation allows applications to be recovered in case of any failures. To equip an application with elasticity and remediation capabilities, multiple techniques have been introduced. Out of those, three techniques are widely applied in practice: load balancing, application scaling, and migration.

Load balancing addresses the effective dissemination of workload coming to a system or an application throughout the available system resources, so as to achieve better efficiency and improve the application throughput. It is a multi-stage process including the identification and allocation of proper resources, and the actual dispatching of the incoming workload. Resources to be allocated could be an instance of an application or a component, a container, a VM, or even a physical machine. As overloading leads to a degradation of the application's performance, load balancing is required for distributed applications to maintain its efficiency and QoS requirements [73]. Load balancing schemes are typically implemented using either static or dynamic balancing method. While the workload placement rules are predetermined in a static scheme, in a dynamic one the workloads are dynamically processed and assigned at runtime. Although static schemes are simple and typically more stable, they do not perform optimally in heterogeneous distributed applications subject to unpredictable workload fluctuations as well as to the variation of workload distribution over time [21, 28, 46].

Scaling (or autoscaling) is an essential technique for cloud computing because it advocates the key principle of *pay-as-you-go* in resource usage on the computing platform. Scaling empowers resources to be allocated or released dynamically according to the demand of the applications, thereby reducing the resources cost due to over-provisioning while maintaining a given QoS. Scaling is now widely employed in the products of major cloud service providers such as Google Cloud Platform, Rackspace, Amazon Web Services and Microsoft Azure. Depending on the resource provisioning methods, scaling can be horizontal (scale out/in) or vertical (scale up/down). To horizontally scale an application means to add/remove a number of instances of a component, i.e., adding/removing VM/component instances on which components are hosted and running to/from the application. Typically, load balancing accompanies this type of scaling in order to fairly distribute the workload among different instances. Vertical scaling works by adjusting the amount of resources, such as computation and/or memory resources, allocated to a running instance. According to the characteristics and requirements of applications, scaling tasks can be performed in either a reactive or a predictive manner. Reactive scaling is triggered to react to the changes under current conditions of the application, whereas predictive scaling first analyzes historical workload data, creates a workload model, uses the model for future workload prediction, and finally generates early resolutions to proactively allocate resources needed by the application in the future. Although predictive scaling methods are more advanced, the reactive ones are widely adopted in commercial products due to the simplicity, accuracy, and high performance [7, 76, 79, 96].

3.3 Infrastructure Optimization

As discussed above, a distributed application consists of multiple components which may come with different sets of requirements. Specifically, components designed to perform CPU-intensive tasks definitely demand a large amount of computational resources. Those dealing with large amounts of data require a high data throughput and a large storage capacity. Other components interacting with the end users are expected to be responsive in communication-intensive tasks. The problem of deploying such different types of component in edge-cloud environments becomes crucial and more challenging due to the diversity in capacities and locations of resources in the cloud, fog, and edge networks. Tackling this problem is apparently coupled with edge-cloud infrastructure optimization. A key technique often used in this context is to leverage the level of indirection offered by virtualization technologies to construct software-defined infrastructures (SDIs) based on VMs and lightweight containers, where components can be dynamically assigned to and migrated between physical resources.

Being built on top of virtualization technologies, SDI provides a full abstraction to cloud DC infrastructure. This implies that cloud resources are softwarized and dynamically programmable, configurable and controllable without any intervention by human operators [95]. In this way, SDI is capable of providing resources flexibly for various service providers' demands in the deployment of their applications/services. Such resource provisioning on demand helps exploit resources effectively and reduce energy consumption. In other words, SDI allows infrastructure providers to optimize the utilization of infrastructural resources and communication networks. Another observable advantage of SDI is to improve the applications' elasticity. In fact, the abstraction of infrastructural resources enables applications to be quickly autoscaled by the initiation of new VMs/containers or via the reconfiguration of existing ones hosting instances of applications/components. Moreover, it permits workload migration, which is impossible or requires too much effort to be automated with physical resources in legacy infrastructures.

Virtualization also facilitates and eases the placement of multiple VMs/containers in a single physical server, which is known as *server consolidation*. Its primary target is to obtain an optimal placement where the optimality criterion depends on the service provider's requirements (e.g., low power consumption) or on the application requirements (e.g., fast response time, redundancy, etc.). Server consolidation relies on application migration techniques which actually migrate VMs/container between physical servers in order to implement the optimal placement solutions. It is especially necessary in the cases that workloads arriving at components are unpredictable [26]. In fact, to maintain the QoS whenever workload variations are experienced, VMs/containers hosting the corresponding components need to be resized/reconfigured and migrated to proper physical servers. This causes workload redirections, which in turn result in consolidations of the workloads of co-hosted application components on specific physical servers. In this way, server consolidation can minimize the total number of physical servers required to host VMs/containers, thereby optimizing the utilization and efficiency of individual resources and clusters. Specifically, hardware costs, operation and administration costs are all minimized thanks to the use of a small number of physical servers.

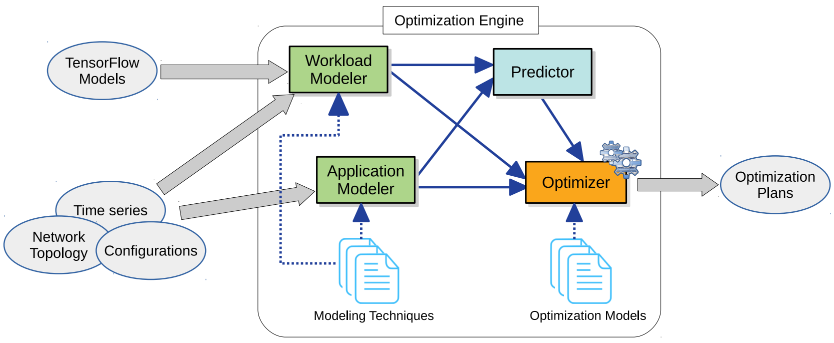

Fig. 4. An example of an optimization engine for VM/container placement. The optimizer is designated to produce the application and/or infrastructure optimization plans of VM/container placement.

3.4 Summary

Fig. 4 demonstrates a general architecture of an optimization engine which is employed to solve problems related to reliable resource provisioning for large-scale distributed applications operating in edge-cloud environments. The problems can be regarding application optimization or infrastructure optimization, and tackling each of them may requires some understanding or models of both the workload and the architecture/structure of the applications. Therefore, the engine is often designed with two *modelers* for workload and application modeling, a *predictor* for producing predictions of workload and application behaviors, and an *optimizer* for solving optimization problems. The workload modeler can take two different types of input: the models generated by third-party modeling tools or the raw data, and then produces workload models to feed the predictor. To support different types of applications and input data, multiple modeling techniques/methods can be used by the two modelers. In addition to the workload models, the predictor also utilizes application models to facilitate its predictions with the workload propagation or the workload distribution throughout the components of the applications. In the same manner, multiple optimization methods/schemes are installed in the optimizer to enable the engine to deal with various problems in different cases and/or with different input data which are provided by the modelers and predictor. The outputs of the optimizer (also the optimization engine) are the optimization plans of VM/container placement.

This survey is motivated by the complexity of the problem of reliable resource provisioning for edge-cloud applications. To the best of our knowledge, there have been numerous studies surveying separately related sub-problems raised in each of the aforementioned topics, but none of them has investigated the problem as a whole to provide a complete technical reference to both academic and industrial research. By way of contrast, our survey sketches a full set of such sub-problems, and then carries out an exploration of the state of the art of machine learning-based techniques to solve them, before discussing how to address the problem as a whole.

4 LITERATURE REVIEW

Fig. 5 depicts a taxonomic classification of the literature surveyed by our paper. The studies have been classified in four main groups. Workload Analysis and Prediction deals with the task to mathematically model workloads and forecast future values given past behavior. Placement and Consolidation is about finding the optimal location of services and applications in large DCs. Elasticity and Remediation cover load balancing and how to properly scale applications. Finally, Network Function Placement studies similar issues in a network function virtualization context. We emphasize that, unless otherwise stated, the works surveyed in the following often cover techniques that can be applied to different virtualization technologies (e.g., VMs and containers) alike. For each paper, we still describe the context chosen in that specific work, and for techniques that can be applied only in a specific context (e.g., live VM migration) this is stressed upon the first occurrence.

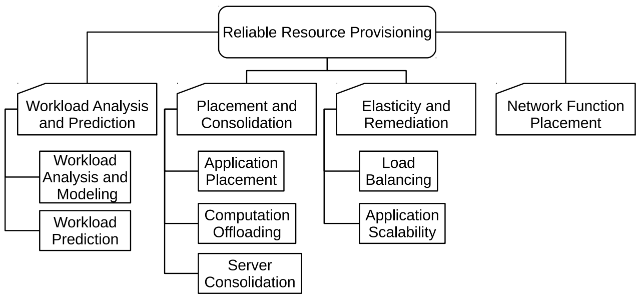
Reliable Resource Provisioning
Workload Analysis and Prediction
Placement and Consolidation
Elasticity and Remediation
Network Function Placement
Workload Analysis and Modeling
Workload Prediction
Application Placement
Computation Offloading
Server Consolidation
Load Balancing
Application Scalability

Fig. 5. Taxonomic classification of the studies reviewed in the survey.

4.1 Workload Analysis and Prediction

Workload analysis and prediction has recently become an important research topic, as testified by the significant body of literature and by the presence of a few surveys covering aspects of this field. The recent work in [10] classifies workload traces collected from various types of applications (web, mobile, video streaming, etc.) as well as from the utilization of the underlying network infrastructure. The work discusses distinct workload features, characteristics, models and techniques to derive them. The models are applied in diverse contexts, in order to facilitate diverse resource planning and provisioning tasks, video content delivery, and system performance evaluation. Given a broad concept of workload and goals, the insufficiency of workload datasets for public use, and the fast evolution of cloud technology, the papers covered in [10] are shown to have different degrees of maturity (the highest being found in web applications and social networks).

Another recent survey [39] explores statistical and machine learning-based methods used to forecast the future workload of online systems given past values and additional features. The survey covers time series analysis methods including autoregressive integrated moving average (ARIMA), generalized autoregressive conditional heterokedasticity (GARCH), and self exciting threshold autoregressive moving average (SETARMA) models, as well as modern machine learning methods including advanced neural networks, support vector machines (SVM), decision trees, Bayesian networks, splines, and exponential smoothing. The selected studies in the survey cover data preparation, workload modeling and prediction methods. Notwithstanding this, most of them exclude up-to-date examples on how to use the methods to forecast future workload.

As the above surveys suggest, workload modeling and prediction in general was born mainly in the context of classical cloud computing scenarios. However, its great potential for the optimization and self-management of complex cloud/fog/edge scenarios makes it important to picture the most employed characterization techniques. This will be the focus of the next paragraphs and will be summarized in Section 4.1.3.

4.1.1 Workload Analysis and Modeling. Jia *et al.* propose to use principal component analysis to find a set of crucial metrics out of a set of 45 metrics to characterize an array of 32 big data workload traces retrieved from BigDataBench [45]. The authors apply k-means clustering and a Bayesian information criterion both to study the affinity of the workloads, and to select a subset of representative ones. The work considers two different software stacks: Hadoop and Spark. These stacks are shown to have a greater impact on the behavior of applications, from a microarchitecture point of view, than the employed algorithms. An alternative approach to characterize application workload is introduced in [50], which proposes to collect hardware performance data for modern processors, as this method is potentially less invasive than traditional monitoring tools. The methodology has been evaluated using two different ARM-based mobile phones, with three applications running in each device.

The authors of [4] develop a tool for low-level workload characterization using data collected from a hypervisor located in the virtualization tier and management operations of VMs. This makes it possible to characterize applications based on a set of typical CPU, memory, storage, and network utilization metrics. To do so, the authors apply a linear combination of various "*canonical workloads*", which correspond to the set of above metrics, coupled with the least absolute shrinkage and selection operator (LASSO) method. The methodology is evaluated through a set of standard benchmarks such as MySQL and Apache web in a lab environment using synthetic datasets. The advantage of monitoring the hypervisor for workload data collection instead of each VM yields a lower footprint on machine performance.

The study presented in [107] has the advantage of using real workload data from a Netflix production server for workload analysis and modeling. The authors illustrate the spatio-temporal characteristics of the user behavior in the workload, through the concepts of chain and phase. A chain denotes a series of requests issued by a particular user to the same Netflix resource, whereas a phase represents a short transient, stable, or inactive period in a session. By subdividing data into chunks according to the above concepts and by analyzing each chunk, the authors obtain fine-grained workload information. To prove the efficiency of their method, the authors develop workload-specific prefetching algorithms that reduce the utilization of storage and system memory. The work in [122] utilizes basic distribution analysis and data decomposition techniques to analyze the memcached workload in a Facebook production system, so as to obtain insight into the role and efficiency of the caching system and the patterns of user behaviors. The results serve to construct realistic key-value cache models and synthetic workloads, and provides directions to structure and design key-value cache systems. The authors of [56] perform an analysis on workload data collected from a virtual Content Delivery Network (vCDN) of an European telecommunication company. Different frequencies are used to decompose the workload data so as to identify workload characteristics and behavior. Next, the authors adopt seasonal ARIMA model combined with Box-Jenkins method to construct the workload models, which in turn are used for workload predictions.

The models provide predictions with root mean squared error (RMSE) at about 0.015. Aiming at container workload characterization, in [16] descriptive analytics is employed to analyze workload traces of Alibaba's DCs so as to highlight the main issues faced by a data center hosting both production-scale long-running jobs as well as more dynamic non-containerized batch jobs. The study highlights typical resource overbooking patterns, and confirms a reliable prediction of resource usage over time is needed to avoid, e.g., straggling issues and co-location interference.

The authors of [119] carry out a comprehensive workload analysis and modeling for the workload data collected from three different private cloud production deployments: a large-scale software development system, a generic cloud, and a business process integration framework. They compare the capability of the hyperexponential, lognormal, and Pareto parametric models to answer "what-if" questions for load variation prediction, infrastructure capacity planning, and cost estimation. The workload of the VMs is characterized in terms of the interarrival time of the requests, VM lifetime, number of VMs requested, and the size of each VM. However, it remains unclear if physical system parameters (CPU, disk, and network usage) can be predicted from the features used for workload characterization, and if the considered three-phase hyperexponential distribution generalizes to all kinds of environments or remains specific to the studied dataset.

The use of system log analysis to automate workload characterization is proposed in [3]. With the results of the analysis, the authors derive sequences of interactions with the system (termed sessions) and propose a customer behavior graph model to reverse-engineer high-level application scenarios at the user interface level, typically in terms of usage patterns. The validity of the model is tested with artificially generated workload for a web server in a testbed. The proposed method is apt for e-commerce systems, which naturally employ a concept of user sessions. In [19], Cuzzocrea *et al.* propose to collect time series of references mapping a program to virtual memory at run time, and to apply spectral analysis to the collected data. The authors then model the time series using an ergodic continuous hidden Markov model in order to characterize application workload, classify applications (e.g., for benchmark construction), and generate synthetic workloads. Experimental validation is carried out using a collection of typical applications.

With the objective of extending CloudSim, [65] presents a workload modeling approach encompassing data analysis, model parameter estimation, simulation and validation. Synthetic workload data used in the study is generated via RUBiS [13]. The collected workload metrics include the CPU, memory, and storage utilization, the system response time, and the total number of executed instructions. Once validated, the models are implemented in CloudSim for public use. The authors of [102] propose three different methods (linear, weighed, and exponential) to model the CPU workload of the control plane of a Software-Defined Network (SDN) hypervisor. The models are tested using an emulated data plane with the FlowVisor and OpenVirteX hypervisors. The study in [103] extends [102] by considering the fluctuation of resource availability, by including a better workload model based on SVM, and by testing the proposed method in a simulation environment.

4.1.2 Workload Prediction. A Bayesian approach is discussed in [22] to predict the future CPU and memory workload of a given application. The authors propose a set of metrics that can be used as predictors (e.g., mean load, fairness index) and their combinations, and evaluate their results against other moving average and autoregressive models via the Google dataset [90]. Similarly [48] proposes to predict the CPU workload on VMs, by joining time series autocorrelation measurements and similarity clustering. The results are validated using a data captured from a real private cloud. A significant improvement in prediction accuracy is observed.

Liu *et al.* present a workload analysis and prediction using trace data of a cluster in the container-based Google cloud [60]. Starting from the duration and waiting time of computing jobs, the study addresses the impact of the correlation between the job state (*submission*/*finish*/*failure*) and the number of machines on the overall cloud performance. After observing that none of the considered prediction methods performs efficiently, the authors propose a weighed ensemble method, where weights are adjusted based on the error time series of the methods.

Hu *et al.* propose a distributed system to monitor and predict the grid resource utilization [40], where prediction is delegated to neural networks and SVM, whose hyper-parameters are optimized through genetic and particle swarm algorithms. The models are tested using historical CPU and network bandwidth datasets, whose relatively small size constrains neural networks to small structures, and may not translate to big data learning on complex cloud architectures.

In [123], autocorrelation is used to identify relevant workload features to train neural networks and forecast future CPU, memory, storage, and network workload. The methodology is evaluated using a real dataset from an IBM production cloud. The work in [126] proposes to train multilayer perceptrons using application workload data clusterized by CPU usage. New tasks are classified based on proximity to one of the available clusters. The methodology is evaluated using the Google dataset. A method to predict power consumption in DCs over different time scales is proposed in [59]. First, the time series are de-noised using a de-trended fluctuation analysis, then power consumption is analyzed at the desired granularity using autoencoder neural networks. The model is validated by integrating real web server workload datasets in a simulated DC.

A combination of theoretical models, neural networks, SVM, and random forests is proposed in [23] to predict future workloads, with the goal of reducing the machine learning training time and increase the accuracy of theoretical models. The authors combine the models through kNN, hybrid boosting, and probing, and evaluate the results in a testbed with two open source applications. Given the small datasets, it remains unclear if the training would be an issue in modern machines. A joint feature selection, clustering, labeling and machine learning approach to resource scheduling is proposed in [124]. The main focus of the work is a set of algorithms leveraging multiple machine learning models to improve the classification and prediction of network nodes' performance. The algorithms are evaluated in terms of performance classification and straggler mitigation using the OpenCloud dataset. The authors report an achievement of up to a 92.86% node performance prediction accuracy using the appropriate algorithm and hyperparameter choice.

In [130], the authors employ a Deep Belief Network (DBN) to make long-term (day scale) and short-term (hour scale) predictions of future CPU and memory requirements in a cloud environment. The results are validated using the Google dataset, and compared to an ARIMA model. The authors discard *recurrent neural networks* (RNNs) due to the long training time. The error achieved by the DBN seems to indicate a risk of overfitting. A 3-layer neural network is trained using genetic algorithms and applied to workload predictions in cloud DCs in [54]. The proposed training method produces a model that predicts the NASA and Saskatchewan servers' HTTP trace data with a higher accuracy then achieved with the backpropagation algorithm.

Workload management is addressed in [67] via decision trees re-trained using reinforcement learning, and applied to the best placement of new database queries. The authors implement and test a proof-of-concept implementation. In the context of automatically scaling computing resources in container-based cloud platforms, neural networks, Q-learning and a custom heuristic algorithm are compared in [97] using synthetic workloads. The authors conclude that Q-learning shows the fastest adaptation rate, and provides up to 22% resource saving.

4.1.3 Summary. Workload analysis presents different levels of maturity, mostly due to the lack of publicly available datasets and to fast pace of cloud technology evolution [10]. Moreover, the software used during the evaluation could have a heavier impact on workload characterization than the selected machine learning algorithm [45]. Further analysis involved the hardware [50], hypervisor [4], operating system [3], middleware [119], application [56, 107], network [102], and a combination of the above [65]. For workload prediction, most of the machine learning alternatives have been evaluated, including Bayesian approaches [22], clustering [48], moving averages [60], SVM [40], decision trees [67], neural networks [54, 123, 126], and combinations thereof [23], without a clear winner [39]. However, neural networks seem to provide significantly good accuracy in several cases, provided that sufficient data is available for training.

4.2 Placement and Consolidation Optimization

4.2.1 Application placement. Application placement deals with the deployment of application components in physical servers, VMs and/or containers, and encompasses different problems such as replica or content placement in CDNs, and placement of components in intra- or inter-DCs scenarios. Each problem is formulated with its own set of constraints on different Key Performance Indicators (KPIs), including resource utilization and the quality/availability of services. Due to the diversity and complexity of the application architecture, deploying applications on different computing platforms requires various placement schemes. In [106], the optimal placement if VM is achieved by solving a linear program, along with simpler heuristic algorithms from the bin-packaging domain. Additional requirements from a managerial point of view are also included. The approach is validated using datasets from an industrial environment and experimental results indicate an order-of 30% cost savings.

Wang *et al.* employ graph-to-graph mapping to solve the component placement problem in order to balance the workload of different components [117]. The authors model both the application and the physical computing infrastructure as graphs and find the optimal mapping between the two graphs. They then propose two different heuristic algorithms to solve the problem under different constraints, and theoretically prove them to achieve a polynomial-logarithmic approximation ratio. In [5], graph-to-graph mapping is used for multi-component application placement in order to minimize both the total mapping cost and the rate of mapping failure, constrained by the network capacity and service delay requirements. After solving the mapping through an integer linear programming, the authors deploy two heuristic algorithms. Through comprehensive simulations, the centralized algorithm is shown to yield a near-optimal result, although its extremely high complexity makes infeasible in large-scale cloud networks compared to the distributed one.

The placement of applications in mobile cloud networks is addressed in [111] via a multi-objective optimization accounting for resource utilization, service latency, and provisioning costs. Besides a globally optimum solution obtained via exhaustive search, the authors devise a local search algorithm based on depth-first search over a local subgraph. A method to optimally allocate of physical machines in cloud DCs is proposed in [6]. Jobs to be scheduled are characterized by required CPU and memory, and an optimal allocation is obtained via linear programming. An approximate solution is computed via a decision tree and linear regression. The proposal is validated experimentally in a testbed with web server jobs. k-means and density-based clustering are used in [125] to aggregate applications with complementary requirements on the same physical machines. The applications are classified as CPU-intensive or non-CPU intensive based on the CPU rate and utilization, and on the cycles per instruction, where non-CPU intensive application show higher disk I/O time and disk usage. The methods are applied to the open Google workload traces dataset.

4.2.2 Computation Offloading. A deeper investigation on component placement shows that the migration of computation-intensive tasks across edge-cloud DCs and mobile devices may lead to significant application performance improvements. Seeking such an improvement leads to the formulation of a *computation offloading* problem, whose objective is to improve the placement of computation-intensive components typically by moving them out of resource-constrained devices [24, 49]. In edge-cloud computing, these components can be deployed either on a mobile device, in a cloudlet, or in an edge- or cloud-DC. Because mobile devices are resource-constrained, placing the components on an edge-cloud server will release the devices from intensive computations, thereby improving the performance of applications and also the user experience [62]. However, offloading incurs communication costs and delays, and different locations may be more or less suitable for a certain target. Therefore, offloading may achieve different benefits due to different levels of server performance or reliability of servers and the communication medium.

Computation offloading is used for better energy efficiency in edge-cloud computing in [120]. The work considers both two-level offloading (where parts of a mobile device's workload is shifted to the cloud) and three-level offloading (where the workload is initially shifted to a cloudlet, and only after that to the cloud). The authors solve the problem with a dynamic control algorithm employing Lyapunov optimization so as to optimally reduce the energy consumption while maintaining the responsiveness of applications. The authors of [34] consider hybrid fiber-wireless networks as a computing infrastructure where the cloud and MEC coexist, and provide offloading opportunities. Offloading is optimized to maximize the energy savings of mobile devices in this context, while satisfying communications and latency constraints. Different approaches include: an exhaustive search; a centralized greedy heuristic scheme; and a game-theoretic scheme that operates in a distributed manner. The work in [112] aims at optimizing computation offloading and resource allocation in MEC to minimize the energy consumption under bandwidth, compute resources, and latency constraints. The study covers offloading both to the cloud (assumed to have unlimited resources), and to resource-constrained mobile peers. The authors show that such problem is non-convex and NP-hard, and solve it iteratively using a successive convex approximation approach.

4.2.3 Server Consolidation. Both the optimization of application placement and the offloading of computationally-intensive tasks bring benefits by reducing the ineffective allocation of resources and by improving the service performance. Still, effective optimization solutions need to also take into account the conditions of the underlying infrastructure, besides pure application-level requirements. Server consolidation with the help of VM migration techniques is a prerequisite to make truly optimal decisions. A survey of VM-specific migration schemes in cloud DCs is introduced in [1]. The work first identifies challenges entailed by the migration of VMs across Wide Area Network (WAN) links. Then, the surveyed schemes are characterized based on several qualitative variables. Next, the authors propose a novel taxonomy to categorize the analyzed schemes and frameworks, and finally list challenges and research trends.

A migration framework is presented in [25], which employs two levels of software agents to find the optimal VM migration time and placement. On each physical machine resides a first-level agent that is trained through Q-learning to detect overloading. A second-level agent decides where to migrate VMs. The framework is simulated using CloudSim and compared with classical migration algorithms. By observing that VMs share base images of common operating systems or software stacks that remain unchanged over time, the CBase migration framework is implemented in [128] using a central repository of common base images. With CBase, a VM migration job consists of a base image, user data and memory migration, where the former two take place concurrently, followed by the latter. The authors enhance the scheme with a multilayered VM structure [127] leveraging the reuse of base images. The consistency of user data migration is ensured via snapshotting.

The authors of [20] observe the bandwidth contention between the execution of migration tasks and VM applications, and propose a traffic-sensitive live migration scheme for co-located VMs that exploits both pre-copy and post-copy. The degree of contention is estimated through a network traffic monitor that collects traffic data from each VM, and the migration sequence that causes the least contention is adopted. The proposed scheme is implemented on KVM/QEMU. Another live migration scheme is presented in [108], which proposes to migrate VM serially with multiple consecutive migration tasks: the first one using the pre-copy procedure and the remaining ones using post-copy. An m-mixed strategy is also proposed to pre-copy m VMs in parallel before serially migrating the remaining VMs using post-copy. A theoretical analysis based on multi-server queues enables the evaluation of different metrics for each migration. The authors conclude that it is feasible to implement the proposed strategies using Xen or the KVM hypervisor. Considering the VM migration over WANs, the approach proposed in [2] employs predictive analytics to reduce the downtime of live migrations. A regression tree is trained to decide the resource utilization thresholds after which a migration should occur. A set of heuristics based on the cost of migration (in terms of the number of CPUs or network usage) combined with the prediction of the VM's future behavior is used to finally decide whether to migrate. The system improves itself as new data becomes available, and is evaluated via a simulated testing environment.

In [110], the authors propose an architecture and algorithm for container migration focused on mobile services and fog computing in large-scale environments. Migration costs are measured via communication delay and power consumption. The algorithm is based on a Markov decision process enhanced with a four-layer neural network and trained with a reinforcement learning strategy. Experimental validation in a testbed shows a 48.5% save in power consumption, and 59.5% reduction in total migration cost with respect to Q-learning.

The survey in [115] introduces a hierarchical classification of server consolidation techniques in cloud DCs, and then discusses their problems and challenges. The surveyed work is organized according to the operational objectives and costs of DCs, and cover different types of optimization methods aimed at addressing server consolidation and energy efficiency. Furthermore, another survey presented in [43] covers resource reconfiguration, primarily focusing on the adaptation of resource assignment in cloud computing, and includes references to fuzzy logic methods. A novel definition of *cloud adaptation* is proposed, which separates the decision making process from the enacting of the reconfiguration. A set of key features used to classify research directions are derived, identifying open challenges.

PRESS is presented in [30] as an approach to predict the CPU usage workload of VMs and to decide the optimal resource-saving VM reconfiguration that still maintains service-level objectives. The approach seeks repetitive patterns through Fourier analysis, and resorts to a discrete-time Markov chain model if no pattern is found. The results are obtained in a virtual lab using synthetic data from RUBiS and real traces from Google, and compared against simpler approaches. The scheme in [70] predicts the future CPU usage of VMs through Brown's quadratic exponential smoothing, complemented with a genetic algorithm to find a VM configuration that can optimally manage the predicted workload. The performance of the proposed scheme is finally evaluated in a sizeable environment with more than 300 servers using artificially generated loads.

The authors of [57] propose to apply a binning algorithm in order to consolidate VMs hosting players of a popular massive multiuser online game. The authors collect time series of the in-game user count data for 270 days, and employ correlation arguments to predict the number of users across subsequent time intervals. The algorithm is evaluated using a simple simulation based on Gaussian user arrivals. Workload variability can be a crucial variable to decide whether to migrate a VM. Based on the rationale that machines facing a steady workload should not be migrated, in [26] the authors propose linear programming-based and heuristic algorithms to optimize VM migrations. The algorithms are validated in a simulated environment based on data traces from TU Berlin and Google DCs, and shown to reduce migrations with minimal impact on SLA. In [42] the authors employ an ARIMA model to predict future CPU and memory workload from past observations taken over predefined time intervals. The predictions are based on the worst-case forecasted workload in any interval. The migration overhead is also taken into account when moving a VM. The proposed methodology is evaluated in a small testbed using synthetic data.

Recently, advanced machine learning techniques have been widely employed to tackle optimal VM migration. A two-level approach to manage the resource allocation of resources in a cloud environment is detailed in [63], with reinforcement learning by means of an autoencoder neural network at the global scale, along with a Long Short Term Memory (LSTM) neural network at the local level. The models are trained using Google data, and their validity is tested in simulations. In [74], SVM and Q-learning are used to automatically and dynamically select which resources of non-urgent virtual networks should be migrated, and where, with the objective to migrate VMs while satisfying QoS requirements. The method is evaluated through simulations showing improvements over a static resource migration approach and a dynamic one with random selections.

4.2.4 Summary. Optimal application placement is mostly modeled as a graph problem [5, 111, 117] and solved using classical linear programming techniques [6, 26, 106]. It can also be combined with computation offloading in order to increase the application performance and the quality of the user's experience [120] but at the expense of higher communication costs and delays. The problem has been extensively studied in case of mobile services [34] and mostly from the point of view of energy saving [112]. The models are typically based on queuing theory [108] and Q-learning [25, 74], although alternative approaches have been investigated, including decision trees [2], time series analysis [42], genetic algorithms [70] and neural networks [63].

4.3 Elasticity and Remediation

4.3.1 Load Balancing. Load balancing in combination with migration techniques can be considered as one of the key methods to realize system/application elasticity and remediation. In fact, redirecting and redistributing workload can improve the application throughput, thereby reducing the service latency and maintaining the robustness of the application. Moreover, in case recovery and remediation are needed to solve problems with the infrastructure and/or the application, the workload directed to migrated applications or components should be redistributed and balanced.

The study presented in [64] formulates a geographical load balancing problem with a cost function that factors in energy consumption and service delay. The main goal is to identify routing paths conveying workloads to DCs and a number of active servers at each DC, so as to minimize the cost function. Three distributed algorithms are proposed to generate provable optimal solutions. The first algorithm is implemented using the Gauss-Seidel method, while the last two are gradient-based and shown to converge to an optimal solution faster. The performance of three algorithms is evaluated and compared to that of existing ones, showing that the proposed solutions outperform solutions considering only the energy consumption or the service delay. Targeting geographical balancing, the authors of [15] propose a two-timescale resource allocation scheme to minimize the overall network cost, defined to reflect energy consumption. The scheme is implemented as a distributed algorithm dealing with long-term and short-term time scales separately, and considering multiple constraints including the computation capacity of DCs, operational costs, power supply, consumption and storage. The performance of the scheme is compared to that of local load balancing and other geographical balancing schemes. Results show that the proposed scheme can decrease the average network cost up to about 46% using data from production systems.

The work in [93] proposes a hybrid approach to minimize the service delay in edge-cloud computing, using VM migration and transmission power control, to balance the workload of cloudlet servers. First, an optimization problem is formulated based on a mathematical model of the service delay, defined as the task processing time plus the network transmission delay. The authors solve the problem using the partial derivative method and then use the solution to make VM migration decisions between cloudlets.

Recently, many studies have targeted at load balancing on microservices deployed on container-based cloud platforms. The one in [35] proposes a container scheduling technique to address the problem. The technique takes into account the attributes of every single container (e.g., size and resource demand) and the interaction between containerized services when carrying out placement and migration. A particle swarm optimization algorithm is adopted to realize the scheduling technique. The performance of the proposed technique is verified and compared to MOPSO [109] and the spread scheduling strategy widely adopted in microservices, showing that the proposed technique outperforms both, improving the system performance by 20% and 25%, respectively. The work introduced in [80] revisits the problem towards the minimization of the system response time. The authors propose an algorithm named COLBA, which is based on game theory and convex optimization, to solve the problem. The performance of the algorithm is evaluated via simulations and the results show that the system response time improves by up to 41% and 13% against the existing instance-oriented and microservice-oriented balancing techniques, respectively.

4.3.2 Application Scaling. Besides load balancing, application autoscaling plays a crucial role to completely achieve application elasticity and remediation. By flexibly adjusting the amount of resources allocated for applications and/or the number of application instances or components, it is possible to adapt to workload fluctuations, which helps prevent the application from becoming unresponsive or terminating. It is worth noting that after performing the application scaling, the load balancing task should balance the workload at every instance of each application.

Aiming at a proactive dynamic resource provisioning for cloud applications, the study in [9] proposes a system consisting of an application provisioner, a load predictor and performance modeler, and a workload analyzer. A workload model is constructed using historical workload data and is updated continuously at runtime by the workload analyzer using ARIMA model. It is used for future workload predictions, which are then exploited by the provisioner to proactively allocate resources and satisfy QoS requirements. The prediction accuracy is evaluated through real data from Wikimedia's web servers. Liu *et al.* address the problem by an adaptive workload prediction scheme and an autoscaling framework presented in [61]. To deal with workload prediction, the authors first formulate the workload classification as an integer programming problem, and then optimally solve the problem through branch and bound. The obtained solution is used to select an appropriate prediction model which is based on either linear regression or SVM. The performance and accuracy of the scheme are evaluated using real trace data obtained from Google cloud.

The authors of [14] find that the allocation of soft resources (e.g., numbers of connections and threads) has a significant impact on the overall performance when autoscaling *n*-tier cloud applications. They construct a model to generate estimations of the maximum throughput of an application, and then use this model to formulate an optimization problem whose solution is derived from an analysis of both the resource utilization and application-related metrics (latency, throughput and number of threads). The solution is then adopted in a framework to derive an appropriate configuration for every tier in scaling tasks. The framework is finally used to realize an autoscaling system with a resource monitor (for input data), an optimization controller (for solution generation and decision making), and actuators (for VM-scaling and resource allocation).

A study on a joint of load balancing and application scaling problem in cloud environments is carried out by the authors of [86]. The cloud infrastructure is assumed to be clustered, and for each cluster a leader is selected to control application admission, autoscaling and load balancing, and also to collaborate with other leaders to perform global resource management. A model is proposed with multiple operating regimes, among which the optimal one indicates an energy-efficient status of a server without any demand of VM migrations in the upcoming scheduling cycles. It also facilitates evaluations of the efficiency of actions required to bring a server to the optimal regime. Based on the model, a set of algorithms are proposed, which are deployed and operated on both the leader and the member servers, to maintain the largest number of servers operating in their respective optimal regime, so as to achieve overall energy efficiency.

In [88], Persico *et al.* propose an horizontal scaling control system for cloud applications encompassing a monitor, a fitness component, a controller, and an actuator. The system employs a proportional-integral-derivative (PID) feedback control mechanism. The PID controller is enhanced by fuzzy logic to be more adaptive in the presence of the unpredictable and time-varying workloads and resources' demands of cloud applications. In addition, the fitness function can deal with multiple metrics (computational and network capacities) so as to meet the applications' demands for different types of resources. The robustness and the remediation capability of the system are demonstrated through experiments with multiple realistic workloads in an Amazon EC2 environment.

With different focus, the authors of [79] study the impact of database tier on horizontal autoscaling in 3-tier web applications in the cloud environment, and implement an autoscaling simulator using the Queuing Network Model (QNM) and Layered QNM. First, they model the cloud application as a network of queues and analyze the response time, throughput, and the maximum amount of workload which can be handled by the application. Analytical results are then used for autoscaling. Finally, a simulation system is constructed with cloud applications deployed on an IaaS infrastructure and an autoscaler deployed to react to workload variations. The system is used for experiments to prove how such a scaling task affects the scaling decisions of the business tier ad how often SLA violations occur.

Elasticity for containerized edge-cloud applications has recently drawn attention from the research community. One exemplary work is presented in [87] tackling the problem of dynamic resource allocation for deep learning jobs, aiming at minimizing their total execution time. The authors propose a job scheduler, called Optimus, based on an online learning technique to construct performance models for the predictions of the jobs' execution time. The scheduler is implemented and integrated with Kubernetes so as to schedule containerized 'workers' serving the jobs. The results show that Optimus outperforms the fairness-based scheduler [29] and Tetris [32] by up to 63% and 139% in terms of makespan and job completion time, respectively.

The authors of [51] propose a heuristic approach to control the elasticity of containerized microservices. The proposed approach is realized by a Custom AUtoScaler (CAUS) mainly performing two complementary tasks: reactive container instance autoscaling in accordance to the changes of workload, and proactive container instances provisioning based on workload predictions. The approach is validated using synthetic workload under two test scenarios (with increase-decrease and spike workload patterns). Results show that CAUS reduces the duration of resource overprovisioning (in terms of the number of containers) up to approximately 30% when comparing to the strategy of overprovisioning with a fix amount of extra containers.

4.3.3 Summary. Load balancing is formulated as optimization problems in most of the cases, which aim at a minimization of different factors such as energy consumption [15, 64], service or system delay [35, 64, 80, 93], and some other combinations [15]. The recently proposed solutions to these problems, which are surveyed in the present paper, are derived using diverse optimization techniques, including derivative-based [93], gradient-based [15, 64], convex [80], and heuristic optimization [35]. Application autoscaling has been widely investigated with different problems tackled through diverse solutions. Most of the studies propose autoscaling systems that proactively and dynamically allocate resources primarily based on the workload predictions generated by predictive models [9, 51, 61, 87], while the others adopt diverse approaches to address the autoscaling tasks such as queuing theory [79], control theory combined with either queuing theory [14] or fuzzy logic [88], and other heuristics [86].

4.4 Network Function Placement

Network Function Virtualization (NFV) is known as a modern network architecture model based on the technology of server virtualization, that splits network functions into basic building blocks implemented as software components. The blocks are chained to create communication services. NFV differs from traditional architectures in that it runs on top of standard servers or cloud computing infrastructure, thus eliminating the demand for custom networking appliances. Thanks to virtualization technologies, NFV-based services, known as service function chains (SFCs), are easily entitled to scalability, elasticity, multi-tenancy support, fast deployment and configuration. The fast scalability and elasticity in turn enable service providers to efficiently maintain a certain degree of redundancy so as to support remediation for their services. Fig. 6 shows a simple architecture of an SFC with multiple virtualized network functions (VNFs) which can be deployed distributedly in edge-cloud environments.

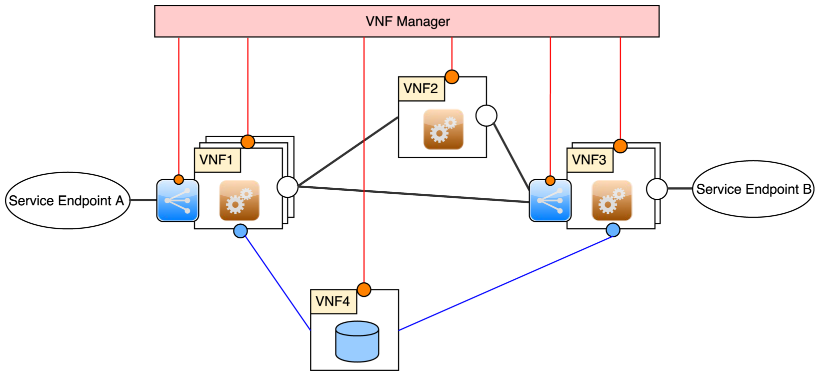

Fig. 6. The architecture of an SFC implemented with three VNFs (VNF 1, 2, and 3), database service and load balancers. The VNF Manager manages and monitors the operations of all the VNFs. VNF1 and VNF3 are scaled with multiple instances independent of other VNFs.

An approach for VNF characterization based on CPU, memory, storage and network resources to be allotted for VNFs is proposed in [91]. The authors model the relationship between the amount of each resource and the achieved performance through a regression tree. The approach is validated through experiments using synthetic data in the context of a traffic classifier VNF which is deployed on top of OpenStack. Its performance is shown to highly depend on the specific characteristics of each network function.

Considering the problem of resource allocating for VNFs in a cloud computing environment, a Markov decision process is applied in [100] to minimize costs while fulfilling predefined QoS constraints. The method is combined with Bayesian learning applied to historical resource utilization data, so as to dynamically predict the reliability of resource provisioning in the future. A performance evaluation is carried out on the proposed method using simulated data. The results are compared to a resource allocation method based on genetic algorithms. The authors of [72] solve the VNF resource allocation problem with an algorithm that employs machine learning to predict future resource requirements for VNF components. Such predictions are obtained through Graph Neural Network model trained with historical and topology information (i.e., behavior of neighboring components). The algorithm is validated under the context of an IP multimedia system optimization. The testbed used for the validation is based on the OpenStack and Clearwaters middleware. Synthetic calls are generated according to a Poisson distribution, and combined with real VoIP data collected from a university. The experimental results are compared to other policies (static and manual) and other machine learning techniques. A machine learning-based autoscaling engine which aims at scaling detection and decision-making tasks in NFV-based systems is proposed in [11]. For scaling detection, several machine learning techniques are applied to realize multiple classification models. The model training process relies on diverse historical data at both the (VNF) application- and infrastructure-level to achieve insight into the systems' behaviors, thereby improving the detection accuracy. The relationship among VNF instances of SFCs is also exploited to compute a plan for flexible resource assignment based on outcomes of the scaling detection. The proposed methodology is evaluated using synthetic workloads for two VNFs: a Squid-based proxy and a Suricata-based intrusion detection. In both cases, the neural network model yields the best results.

The work in [89] proposes a service chaining algorithm for VNFs. The applied method addresses main limitations of competing algorithms including the large convergence time and the constraint that the network configuration should remain static. The proposed algorithm employs genetic programming to generate a near-optimal solution two-order of magnitude faster than other linear programming algorithms. Additionally, the algorithm takes into account the changes in both computing resources and networking resources (given the latest advances in SDN). To show the effectiveness of the proposed algorithm, the obtained solution is evaluated using a custom testbed. The authors of [37] focus on simultaneously optimizing the cost and the latency of the placement of VNFs in SFCs. To this end, they propose an optimization framework employing a random selection of clouds and cloud resources combined with a predictive model based on support vector regression. The framework is evaluated using a combination of synthetic data generated using a queue-theoretic model and data gathered from a cloud testbed. The study in [52] proposes a reinforcement-learning based dynamic flow routing algorithm, which selects a path from a set of multiple paths for one SFC in an SDN/NFV environment. This is achieved by considering both the CPU usage of each node and the network bandwidth of the links between the nodes. The performance of the algorithm is evaluated in an artificial environment and compared with a classic greedy algorithm.

To cope with remediation, the problem of detecting and localizing network-related issues is investigated in [36]. The authors propose a predictive classification model to deal with the problem of NFV deployments within a multi-cloud environment. The model is developed based on a neural network model which adopts multiple machine learning techniques including typical "*shallow*" techniques (SVM, alternating decision tree, and random forest) and deep ones (the stacked autoencoders). In this model, the detection checks if any issue occurs, and then determines the issue type using a shallow learning. Next, with deep learning, the localization locates the issue within the NFV-based system and estimates its impact on the system performance. The proposed model is validated using two different datasets: a real dataset of network faults from the Telstra network, and a synthetic dataset obtained by using a multivariate kernel density estimation technique enhanced with real live network issues.

The concept of Knowledge-Defined Network has been introduced recently in [69], which proposes to add a *knowledge plane* to the planes defined by a traditional SDN architecture. The knowledge plane is designated to exploit advantages of machine learning to address the most challenging problems of complex networks. The authors show four use cases illustrating the application of their proposal and provide solutions for two of them. The first one relates to routing on an overlay network, whereas the other revolves around NFV using real data from a university campus.

4.4.1 Summary. A wide range of problems regarding NFV is investigated, including VNF characterization [91], resource allocation and autoscaling [11, 72, 100], service chaining [37, 52, 89], service remediation [36], and NFV-based knowledge plane addition in SDN [69]. In most of the cases, machine learning techniques are adopted and showcase remarkable outcomes.

5 DISCUSSION

5.1 Analysis and Categorization

Tackling the problem of reliable resource provisioning for edge-cloud applications requires addressing an array of sub-problems including workload analysis and prediction, optimization in resource utilization at both the application and infrastructure levels. Numerous methods and schemes proposed for such purposes have been reviewed in the previous section. Most of them adopt one machine learning technique in their operational flow. Table 1 summarizes the techniques utilized in these methods and schemes. We remark that our paper has adopted a broad meaning for machine learning, which embraces both recent approaches using, e.g., deep neural networks (DNN) or complex graph analysis, as well as more classical approaches including Q-learning, reinforcement learning, Bayesian methods, and Markov models. In accordance with the definition provided in Section 2.5, machine learning methods include all the models that can be trained using data in order to carry out supervised or unsupervised classification and regression tasks.

Table 1. Summary of techniques employed by the studies in the survey.

Applied Technique	References
Regression (Logistic, LASSO, etc.)	[4], [11], [50], [60], [61]
Analytical models (Autocorrelation function (ACF), ARIMA, etc.)	[9], [42], [56], [57], [123], [130]
Bayesian methods	[22], [45], [100], [124]
Markov models	[15], [19], [30], [48], [100], [110]
Queuing theory-based models	[14], [64], [79], [108], [120]
Clustering (K-means, K-nearest neighbors, ...)	[23], [45], [125]
Exponential ensembles	[70], [102], [117], [119]
Artificial neural networks	[11], [23], [36], [40], [53], [54], [59], [60], [63], [69], [72], [87], [97], [110], [123], [126], [130]
Support vector machines	[40], [23], [74], [102], [36], [37], [61], [124]
Decision trees, Random forests	[2], [6], [11], [23], [36], [67], [91], [124]
Reinforcement Learning	[25], [52], [63], [67], [74], [97], [110]
Control theory, Game theory-based techniques	[14], [34], [80], [88], [120]
Linear programming, Geometric programming	[5], [6], [26], [61], [106], [112]
Genetic algorithms, Genetic programming	[40], [54], [70], [88], [89]
Graph analysis	[3], [5], [111], [117]
Distribution analysis	[65], [107], [122]
Derivative-based and Gradient-based optimization	[15], [64], [93]
Agent-based and Heuristic-based methods	[20], [25], [35], [51], [86], [97], [128]

From the present survey, it can be concluded that workload analysis and prediction is a critical task in large-scale cloud- and edge-based computing infrastructures, since high accurate predictions facilitate the dynamic reallocation of available resources to guarantee SLAs meanwhile saving energy and costs. However, workload prediction is in general a difficult endeavor, even more if the workload depends on unpredictable human activities (e.g. on web servers, CDNs, transportation system in smart cities). Classical approaches to workload prediction are based on autoregressive models, like ARIMA or GARCH. Recently, a new generation of data analysis models based on RNNs have been adopted, and extensive evaluations on them show promising results. Neural network models, e.g., LSTM, provide more accurate predictions on average than classical regressive models. Nevertheless, neural networks require an extremely large amount of training data and extensive computing time, and even the use of highly specialized hardware, like servers with GPUs.

The optimal placement of services in cloud environments is a highly relevant issue that has been extensively studied. The problem has reached an unprecedented scale with the recent advent of the fog and edge computing paradigms, where it is often critical to locate services near the user, so as to move computation capabilities instead of moving the data. Finding an optimal solution to the placement of VMs/containers on physical servers is a difficult problem that could become computationally intractable in case of large clusters, edge networks, or networks of IoT devices. Numerous algorithms have been evaluated to find approximate solutions, from the classical linear programming, to more advanced machine learning-based algorithms (decision trees, neural networks, and others). Multiple, and sometimes contradictory requirements can be used as part of the utility functions, and as input attributes to those functions. Making decisions on which parameters to use is a complex research problem by itself that can be addressed by means of applying machine learning techniques like principal component analysis.

Elasticity and remediation are two key functions to fully address the problem of reliable resource provisioning for edge-cloud applications, as they ensure the reliability and robustness of the applications regardless the non-linear fluctuation of the workload over time. However, an optimal implementation of elasticity and remediation in heterogeneous distributed environments with resource virtualization using classic mathematical optimization techniques is computationally challenging. This is because applications running in such an environment are dynamically distributed and highly unpredictable; and each application component may be subject to different workload. According to the reviews in Section 4.3, all the proposed schemes require an understanding of workload characteristics in order to efficiently perform the autoscaling and remediation. Therefore, workload analysis and prediction based on time series analysis becomes the vital factor for the efficiency of the schemes, and especially so for the proactive schemes, as they require future workload forecasting to adjust the amount of resources allocated for applications beforehand. Our reviews also shows that many of studies apply queuing theory and graph theory to model the service latency, which is one of the KPIs that needs to be improved through autoscaling and remediation. Moreover, control theory combined with advanced machine learning optimization techniques has been used for constructing the autoscaling system or framework. As a result, it is worth considering an adoption of the machine learning techniques when deriving holistic solutions to the problem.

Table 2 provides a summary of problems studied by each of the reviewed papers, the tools and methods adopted, and a short quantitative summary of the achieved results. The works have been organized according to the subsections of Section 4, and chronologically within each category. This clearly highlights that the use of complex machine learning and optimization techniques (e.g., deep learning, complex graph theory, and genetic algorithms) relates mainly to the most recent papers.

Table 2. Summary of the proposals, tools/methods, and main results of the studies in the survey.

Ref.	**Proposition**	**Tools / Methods**	**Main results**
		Workload Analysis	
[4]	Using hypervisor performance information to derive application behavior	LASSO regression applied over canonical workloads	A proof that information provided by hypervisors is sufficient to characterize application workload
[122]	Characterization of Facebook's memcached system	Distribution analysis, data decomposition	Insight into Facebook's cache: performance, cache efficiency, cache's key popularity, structure of requests
[50]	Characterization of hardware performance counters of processors	Characterization matrices, Pearson correlation	Derived synthetic workloads reproduce real performance with an average error of 2.8%
[45]	Identifying relevant metrics and select representative workloads	Principal component analysis, K-means, Bayesian information criterion	Software stacks can have more impact on workloads than applications in 80% of the cases

[119]	Investigating the development of parametric statistical models	Exponential ensembles, lognormal, Pareto	3-phase hyperexponential models are more appropriate than lognormal and Pareto: RMSE down to [0.00 − 0.02]
[65]	Web application modeling	Distribution analysis: Generalized lambda, extreme value, Weibull distributions	The proposed model provides high accuracy in data generation: error rate is less than 10%
[3]	Analyzing log-files collected from OS and middleware	Customer Behavior Model Graph analysis	Inferred user behavior model is well matched with the actual workflow of Apache OFBiz
[102]	Estimation of SDN hypervisor resources	Linear, weighted and exponential models	Fast convergence of exponential model (100s) compared to linear models (300s)
[107]	Devising tools for Netflix workload characterization	Distribution analysis	Data prefetching improves hard drive and memory utilization at 13% and 30%
[103]	Estimation of SDN hypervisor resources	SVM	Estimation error is 5.8 % with a standard deviation of 2.5 %
[19]	Analyzing virtual memory references	Spectral analysis, hidden Markov model	Ergodic continuous models gain higher accuracy (76%) in workload classification than discrete ones (65%)
[56]	Analyzing traffic data of a vCDN	Seasonal ARIMA	Achieve low data fitting and prediction RMSE (at about 0.015)
[16]	Characterization of Alibaba's DC workloads	Distribution Analysis	Highlight issues of DCs related to overprovisioning, overbooking, overcommitment, and workload co-location for various workload types
Workload Prediction			
[40]	Modeling and optimization of resource prediction models	Neural network, SVM, genetic algorithm, particle swarm	SVM provides the best results in terms of accuracy and efficiency
[48]	Prediction based on cross-VM workload correlation	Clustering, hidden Markov models	Prediction accuracy is improved from 55% to 76% over methods based on single time series prediction
[22]	Workload prediction for cloud systems using Bayes model	Bayes classifier	Bayes method improves 5.6-50% accuracy over moving-average- and autoregression-based models
[23]	Combination of machine learning algorithms and theoretical models	Neural network, SVM, decision tree	The proposed combination achieves higher prediction accuracy (40% RMSE reduction) than any other ensemble technique
[123]	Using autocorrelation to identify relevant features and forecasting	Neural network, Autoregressive model	Neural networks provide up to 3 times better accuracy than autoregressive models
[60]	An ensemble workload prediction method	Moving average, weighted average, linear regression, neural network	The ensemble of the four listed methods provides higher prediction accuracy than any single method
[67]	Placement of database queries based on machine learning	Decision tree, reinforcement learning	The proposed solution is within 8% of the optimal scheduling cost
[59]	Prediction of power consumption of DCs based on machine learning	Autoencoder neural network, recursive autoencoder	Outperform canonical prediction methods up to 79% error reduction
[126]	Using clusterized data to train a neural network	Clustering, neural network	Up to 50% improvement over existing non-clustered neural networks for small learning rates
[61]	An architecture of adaptive workload prediction system	Linear regression, SVM	Achieve 8% and 29% error reduction compared to SVM and ARIMA
[130]	Automatically learning hyperparameters	Deep belief network, autoregressive model	Achieve 72% error reduction compared with autoregressive models

[97]	Resource prediction for container-based cloud platforms	Q-learning, neural networks, heuristic	Achieve 22% resource saving using Q-learning under a synthetic workload
[124]	Using machine learning models in nodes' performance prediction to improve resource scheduling	SVM, Boosting, Decision Tree, Random Forest, Naive Bayes	Attain a prediction accuracy up to about 93%
[54]	Training a neural network using a genetic algorithm	Neural network, genetic algorithm	Outperform the backpropagation network approach; prediction error (RMSE) reduction down to 0.001
		Application Placement	
[106]	Placement of virtual machines in DCs	Linear programming	Achieve 31% fewer servers with respect to static allocation
[6]	A combination of machine learning algorithms and theoretical models	Decision tree, linear regression	Achieve less than 1% estimation error
[117]	Applying graph-to-graph mapping modeling technique	Graph theory, graph-to-graph mapping, dynamic programming, online approximation algorithms	The proposed solution outperforms typical ones in reducing resource utilization, and its core algorithm achieves an approximation ratio of 2
[111]	Applying searching methods on graph	Graph theory	Up to 25% reduction in cost to run DCs compared to naïve methods
[112]	Offloading computation-intensive tasks from user/mobile devices to both the cloud and other peers	Successive convex approximation, geometric programming	Around 60% and 10% reduction in energy consumption against two models of local computation and cloud offloading, respectively
[125]	Optimal VM placement based on clustering tasks according to resource usage	K-means, density clustering	Density based clustering is faster than k-means (with the rate of 0.17/0.21)
[5]	Applying graph-to-graph mapping modeling technique	Graph theory, graph-to-graph mapping, integer linear programming, heuristics	The obtained result is close to the optimal solution (within 10.7%)
[120]	A mobile cloud offloading system with multiple level of computation offloading	Lyapunov optimization, queuing theory	Around 50% reduction in energy consumption against the local deployment on mobile devices
[34]	An architecture and algorithms for computation offloading in edge-cloud environments	Game theory, greedy heuristics	Obtain near-optimal solutions and 20% improvement (in average) over an existing distributed algorithm
		Server Consolidation	
[57]	Studying server consolidation in domain of massive online games	Autocorrelation functions	Achieve up to 62% energy reduction when applying the proposed technique
[70]	Applying Brown's quadratic exponential smoothing to forecast CPU usage	Exponential smoothing, genetic algorithm	A proof that unnecessarily running machines can be switched off without any system performance degradation
[30]	Optimal reconfiguration of virtual machines	Fast Fourier transform, Markov chain	Prediction accuracy at less than 5% over-estimation error and near zero under-estimation error
[26]	Using variability of workload as a criteria for migration	Linear programming, heuristics	Achieve a significant reduction in the total number of migrations
[42]	Dynamic consolidation of workloads	Elastic ARIMA	Attain 82% values correctly predicted against 52% of classic ARIMA
[25]	Applying multi-agent model in VM consolidation	Agents, Q-learning	Reduce energy costs (6.6% – 36.6%) and SLA violations (10% - 50%) against typical methods
[108]	Adopting multiple techniques to deal with the live migration of multiple VMs	Queuing theory	Lower the downtime extremely in all tests, especially when the number of parallel VM-migrations increases

[2]	Applying predictive analytics to reduce downtime migration	Decision tree, heuristics	Achieve 15% improvement in total migration time over existing schemes
[128]	Live VM migration over WAN with a central base image repository	Heuristics	Improve the migration time up to 19% over traditional and simple techniques
[74]	Resource migration between virtual networks using machine learning technique	SVM, reinforcement learning	The number of time slots with QoS satisfaction is doubled when compared to a static resource allocation method
[63]	Adopting two levels of resource allocation for VMs	Reinforcement learning, autoencoders, recurrent neural network	Greater than 16% improvement in energy saving against a deep reinforcement learning based scheme
[20]	A traffic-sensitive live VM migration scheme	Network traffic profile based heuristics	Lower the migration time up to 49% against typical migration techniques
[110]	Employing machine learning techniques for container migration in fog computing environments	Markov decision process, neural networks, reinforcement learning	Achieve 48.5% power saving and 59.5% migration cost reduction when applying the proposed method
	Load Balancing		
[64]	Load balancing scheme facilitating request routing to DCs and resource allocation in each DC	Queuing theory, Gauss-Seidel method, gradient-based algorithms	The proposed method achieves over 40% reduction in DC operating cost
[93]	Load balancing between cloudlets based on VM migrations	Queuing theory, Shannon-Hartley theorem, partial derivatives	Improve service latency up to 300 ms over conventional approaches, based on analysis and various simulations
[15]	Applying the gradient method and Markov chain to optimize load balancing in cloud DC networks	Markov chain, stochastic dual gradient method	Lower the average network cost up to about 17% or 46% against different existing algorithms
[35]	Microservices load balancing using metaheuristics	Particle swarm optimization	Improve the system performance against existing schemes (20%–25%)
[80]	Microservices load balancing using convex optimization	Game theory, convex optimization	Response time is improved up to 41% and 13% against the existing instance- and microservice-oriented techniques
	Application Scaling		
[9]	A proactive autoscaling approach based on workload prediction	ARIMA	A high prediction accuracy (91%) benefits the resource utilization and reduces QoS violations
[86]	A load balancing and autoscaling model/scheme with energy-efficient awareness	Heuristics	Obtain a near-optimal computational efficiency, up to 3.6 – 3.8 out of 4.46 (transactions/Watt)
[14]	Considering soft resource allocation when autoscaling *n*-tier cloud applications	Control theory, queuing theory, least-square fitting method	The proposed model improves system throughput up to 30% over the default configurations for the testing system.
[79]	An autoscaling simulation package to verify the impact of database layer on 3-tier web service system scaling decisions	Queuing theory	Two hypotheses of how a scaling of database tier impacts the scaling decisions of the business tier and the degree of SLA violations are validated
[88]	A control architecture for horizontally scaling cloud resources	Genetic algorithm, control theory, fuzzy logic	Autoscaling error rate is lower than 30%; the improvement on *integral of squared errors* over other typical and simple schemes is up to a rate of 7/12
[51]	An approach to control elasticity of microservices	Heuristic	Reduce the duration of resource overprovisioning up to 30% compared to simple elasticity strategies
[87]	Dynamic resource allocation for deep learning jobs	Online learning	Outperform typical cluster schedulers up to 63% and 139% in terms of makespan and job completion time

Network Function Placement			
[100]	Applying Markov decision process to allocate NFV components	Markov chain, Bayesian learning	The proposed algorithm can obtain global optimal solution
[89]	A fast chaining algorithm for both computing and network resources	Genetic programming	Attain up to three orders of magnitude faster than traditional solutions
[91]	Characterizing and modeling relationship between resources allocated for VNF deployment	Decision tree	A deployment using Single Route I/O Virtualization brings a 10-fold improvement against Open vSwitch
[37]	Using random search algorithms for fast VNF placement	Random search, SVM	Acceptance rate above 90% in the tested configuration
[72]	Predicting future resource requirements using past historical information and topology	Graph neural network	Enhance processing delay 29% and reduce call drop rate by 27%
[11]	Detecting resource flexing events based on features of VNFs and the NFV infrastructure	Neural network, decision tree, random forest, logistic regression	The neural network provides a gain between 5% and 19% in accuracy with respect to other models.
[69]	Introduction of the Knowledge-Defined Network paradigm	Neural network	Experimental validation with a less than 1% of relative error
[52]	Applying machine learning to routing path selection for SFCs	Reinforcement learning	Obtain a service path closest (with a ratio of 0.8) to the optimal solution
[36]	A multilayer fault detection and localization model based on DNN	Autoencoders, SVM, decision tree, random forest	Achieve 100% accuracy in test datasets with deep learning models

A common recurring pattern that appears in most of the reviewed papers is that the machine learning models proposed have not been trained and tested using large, high-quality datasets gathered from a strong industrial players in production environments. Most of the works are based on synthetic datasets, small datasets, or datasets that are not representative of real scenarios. Only a few publicly available datasets could be considered as good representatives of real production environments. These are the ones published by Google, Facebook, Netflix, and Wikimedia, although the datasets are still comparatively small and old (see Table 3). This lack of relevant and fresh data makes it difficult to correctly assess the quality of the published results, and even more important, to compare the results among competing works. A large, realistic, publicly available dataset from a powerful industrial partner would work as a benchmark, and would allow the scientific community to evaluate and compare different ideas and approaches to a much broader extent.

Table 3. Summary of performance evaluation methods and techniques employed by the studies in the survey.

Evaluation Approach	**References**
Synthetic Dataset	[2], [3], [4], [11], [14], [15], [19], [20], [30], [35], [36], [37], [42], [51], [54], [65], [72], [86], [88], [91], [97], [110], [111], [119], [128]
Historical Dataset	[6], [9], [15], [16], [22], [23], [26], [30], [36], [45], [48], [56], [57], [59], [60], [61], [63], [64], [69], [70], [72], [80], [106], [107], [110], [119], [122], [123], [125], [126], [130], [124]
Simulation	[2], [3], [5], [6], [9], [15], [25], [26], [30], [34], [35], [37], [45], [52], [54], [57], [59], [60], [61], [63], [64], [65], [69], [72], [74], [79], [80], [86], [89], [100], [102], [103], [107], [111], [112], [117], [119], [120]
Testbed	[4], [14], [20], [23], [40], [42], [45], [50], [51], [67], [70], [87], [88], [91], [97], [110], [128]
Production Deployment	[106]

Moreover, it would be highly desirable to achieve a deeper involvement of the industrial players in the studies, not only by providing relevant requirements, but also by contributing to the evaluation of the results, and in particular to the deployment of small-scale pilots in production environments.

Although testbeds make it possible to test different ideas and configurations, without an evaluation on a production environment is difficult to draw final or complete conclusions.

5.2 Recommendation of an Optimization Framework

The survey together with the discussion above motivates us to come up with a framework, based on which solutions to the problem of reliable resource provisioning for edge-cloud applications can be designed, developed, and executed. As illustrated in Fig. 7, the recommended framework entails three different optimization levels for the deployment and management of applications in heterogeneous edge-cloud environments. Each level requires a certain degree of understanding about the applications as well as optimization schemes in resource provisioning. The solutions achieved at each level aim to fulfill different degrees of requirements of various use cases in reality. The core of the framework is an optimization engine which consists of multiple modelers and optimizers used for different purposes. More specifically, the modelers aim to produce different models for each application, including network models, workload models, user mobility models, and application models, among others. All models are used to provide a complete view of the application and as an input for optimizers to deal with a wide range of optimization problems related to the placement, deployment, autoscaling, and remediation of the application.

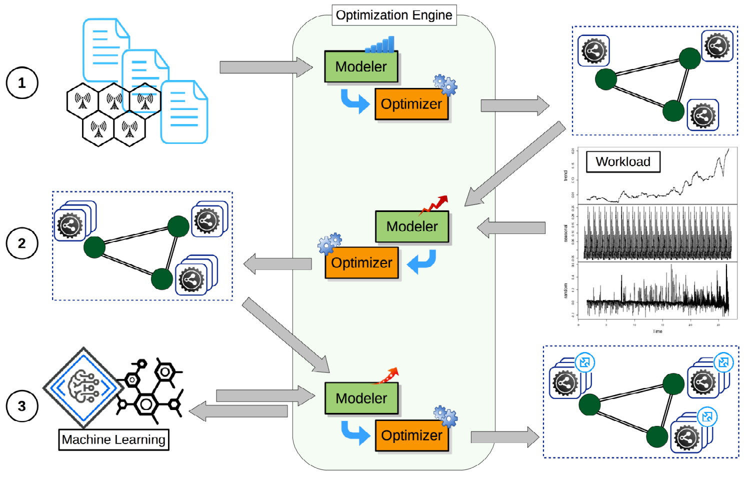

Fig. 7. A stratified approach to constructing an edge computing optimization framework iteratively building on optimization three building blocks: 1: classic optimization on static data, 2: application adaptation variations in workloads and resource availability, 3: joint autoscaling and optimization in multi-tenancy scenarios using machine learning.

As observed in the figure, the first level of optimization is the simplest one which aims at the placement of applications throughout the edge-cloud environments under fixed network, application, and QoS requirements constraints. This has been widely investigated in general, but still needs to be revisited so as to derive specific solutions for every particular use case. Solutions can be used for long-term resource planning or as the initializations for further optimization levels. In the next level of optimization, the variations of workload and user behaviors are taken into account for dynamic application placement and autoscaling. The workload model and user mobility model are used to estimate the demand of resources of every single component for each application. With the estimation results, resources are then allocated properly for each application component or the workload can be redirected or migrated so as to maintain the load balance within an application or among applications. The highest level of optimization aims at proactive resource provisioning for applications. Machine learning is adopted to improve the understanding of both workload and application behaviors, which means more fine-grained models are derived and models can be refined and improved over time. Using these models, predictions can be performed more accurately to support load balancing, autoscaling, and remediation in a proactive manner in real-time.

5.3 Research Trends and Opportunities

In this section we present a number of research trends and opportunities that correspond to open or partially addressed issues in the current literature, and thus constitute promising research directions. We subdivide such directions into five categories: workload distribution and propagation; data generation and data privacy; abnormal workload prediction; server consolidation optimization; and application remediation and reliable service provisioning.

1. **Workload Distribution and Propagation**

For a distributed edge-cloud application, besides understanding workload behavior at every single component, a so-called *workload propagation model* is needed. Such a model represents the knowledge of workload distribution/propagation through or between components. In fact, understanding how a workload variation at a certain component affects that of the other ones is useful for a proactive resource allocation for the entire application. For this reason, studies on standardization of workload propagation models as well as construction of base models at least for some typical systems/applications are crucial.

2. **Data Generation and Data Privacy**

In order to acquire a workload propagation model for a distributed edge-cloud application, workload measurements for every application component are required. Unfortunately, in reality it is impossible and infeasible to collect workload data at every network location in edge-cloud environments where applications are deployed and operating. Hence, there is a need for schemes to derive workload at every application component (using interpolation, extrapolation or generation techniques) from given workload datasets collected at a limited number of locations. A related problem is how to disseminate the collected/generated workload data to the research community in a secure manner so as to protect confidential information. Recently, much attention has been devoted to data privacy. However, an establishment of data privacy for all datasets of large-scale production applications is costly (in terms of both processing time and computational resources) most of the time. One promising approach is to apply data privacy to, and then publicly open only a subset of data together with workload generation models for such dataset. This actually forms a challenging problem for workload data dissemination.

3. **Abnormal Workload Prediction**

It is observable that there typically exist peaks (outliers) in workload data collected from real production systems. An outlier represents a rare event at which an abnormal amount of computational resources is required. Therefore, a failure in addressing outliers likely leads to SLA violations. Although anomaly detection has been widely investigated recently, a prediction of outliers in workload data still lacks systematic approaches in the edge-fog-cloud computing literature. For reliable resource provisioning, such abnormal workload prediction needs to be addressed.

4. **Server Consolidation Optimization**

The recent trends in consolidation optimization research bend towards the combination of workload prediction models and consolidation algorithms; by this combination, it is possible to dynamically adapt the required number of physical machines in advance to satisfy future needs. However, the new combined problem entails further investigation, not only about how to properly combine the independent results of two unrelated families of algorithms, but also by developing mathematical models that could take into consideration all the relevant parameters at the same time. Advanced optimization techniques based on genetic algorithms, genetic programming and other natural solvers have been evaluated in the literature. The development of new flexible software and hardware architectures that can timely deploy the proposed changes is also required.

5. **Application Remediation and Reliable Service Provisioning**

It is challenging to remediate distributed edge-cloud applications because application remediation may require remediations at multiple components. To accomplish application remediation and ensure reliable service provisioning, platforms and/or protocols for components' status exchange and collection are needed. Due to the highly distributed of application components from the central cloud down to the edges of networks, traditional remediation means and approaches applied to monolithic applications or even SOA-based applications running in central cloud DCs are not applicable because of the high communication and/or synchronization costs. Accordingly, remediation for edge-cloud applications is a vital open research topic.

6 CONCLUSIONS

In this survey, we have reviewed how the problem of reliable resource provisioning in joint edge-cloud environments has been addressed in the scientific literature, and in particular what kind of methods have been used to improve the reliability of distributed applications in diverse and heterogeneous network environments. There has been observed, in recent years, a substantial rise in the number of studies that investigate how to apply machine learning techniques to perform characterization and prediction of workload and application behaviors, and also to control complex distributed applications. It is also observable, on average, machine learning techniques provide better results than traditional methods, especially when dealing with sizeable and complex environments.

As witnessed, authors have applied a large number of different mathematical techniques, ranging from classical statistics and modeling to more advanced machine learning algorithms. Future research should be focused not on using yet another mathematical method, but on how to improve over already tested architectures, based on a small subset of the most promising machine learning techniques (e.g. boosted trees or deep neural networks). Moreover, it is required to produce reproducible results, facilitating the comparison of different proposals. In this sense, it would be greatly helpful to have a collection of large and high quality datasets provided by relevant industry players that could be used as benchmarks.

ACKNOWLEDGMENTS

This work has received support from the EU Horizon 2020 programme grant no. 732667 (RECAP), and from the Spanish Ministry of Science, Innovation and Universities grant DisCoEdge (TIN2017-88749-R) and the Comunidad de Madrid grant EdgeData-CM (P2018/TSC-4499).

REFERENCES

[1] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia. 2015. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. *Journal of Network and Computer Applications* 52 (Jun 2015), 11–25.

[2] M. Arif, A. K. Kiani, and J. Qadir. 2017. Machine learning based optimized live virtual machine migration over WAN links. *Telecommunication Systems* 64, 2 (Feb 2017), 245–257.

[3] M. Awad and D. A. Menascé. 2015. Automatic workload characterization using system log analysis. In *Proc. Computer Measurement Group Conference on Performance and Capacity.*

[4] F. Azmandian, M. Moffie, J. G. Dy, J. A. Aslam, and D. R. Kaeli. 2011. Workload characterization at the virtualization layer. In *Proc. IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*. 63–72.

[5] M. Barshan, H. Moens, S. Latre, B. Volckaert, and F. D. Turck. 2017. Algorithms for network-aware application component placement for cloud resource allocation. *Journal of Communications and Networks* 19, 5 (Oct 2017), 493–508.

[6] J. L. Berral, R. Gavalda, and J. Torres. 2011. Adaptive scheduling on power-aware managed data-centers using machine learning. In *Proc. IEEE/ACM International Conference on Grid Computing (GRID)*. 66–73.

[7] A. Biswas, S. Majumdar, and B. Nandy. 2015. An auto-scaling framework for controlling enterprise resources on clouds. In *Proc. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. 971–980.

[8] R. Boutaba, Q. Zhang, and M. F. Zhani. 2013. Virtual machine migration in cloud computing environments: Benefits, challenges, and approaches. In *Communication Infrastructures for Cloud Computing* (1 ed.), H. T. Mouftah and B. Kantarci (Eds.). IGI Global, Hershey, PA, USA, Chapter 17, 383–408.

[9] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya. 2015. Workload prediction using ARIMA model and its impact on cloud Applications' QoS. *IEEE Transactions on Cloud Computing* 3, 4 (Oct 2015), 449–458.

[10] M. C. Calzarossa, L. Massari, and D. Tessera. 2016. Workload characterization: A survey revisited. *ACM Computing Surveys (CSUR)* 48, 3 (Feb 2016).

[11] L. Cao, P. Sharma, S. Fahmy, and V. Saxena. 2017. ENVI: Elastic resource flexing for Network function VIrtualization. In *Proc. USENIX Conference on Hot Topics in Cloud Computing (HotCloud)*. 11.

[12] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguadé. 2012. Autonomic placement of mixed batch and transactional workloads. *IEEE Transactions on Parallel and Distributed Systems* 23, 2 (Feb 2012), 219–231.

[13] E. Cecchet and J. Marguerite. 2009. RUBiS: Rice University Bidding System. (Oct 2009). Retrieved Mar 11, 2018 from http://rubis.ow2.org/

[14] H. Chen, Q. Wang, B. Palanisamy, and P. Xiong. 2017. DCM: Dynamic concurrency management for scaling n-tier applications in cloud. In *Proc. IEEE International Conference on Distributed Computing Systems (ICDCS)*. 2097–2104.

[15] T. Chen, A. G. Marques, and G. B. Giannakis. 2017. DGLB: Distributed stochastic geographical load balancing over cloud networks. *IEEE Transactions on Parallel and Distributed Systems* 28, 7 (Jul 2017), 1866–1880.

[16] Y. Cheng, Z. Chai, and A. Anwar. 2018. Characterizing co-located datacenter workloads: An Alibaba case study. *arXiv preprint* (2018). http://arxiv.org/abs/1808.02919

[17] M. Chiang and T. Zhang. 2016. Fog and IoT: An overview of research opportunities. *IEEE IoT Journal* 3, 6 (Dec 2016), 854–864.

[18] A. Choudhary, M. C. Govil, G. Singh, L. K. Awasthi, E. S. Pilli, and D. Kapil. 2017. A critical survey of live virtual machine migration techniques. *Journal of Cloud Computing* 6, 1 (Nov 2017), 23.

[19] A. Cuzzocrea, E. Mumolo, and G. Vercelli. 2017. Ergodic hidden Markov models for workload characterization problems. In *Proc. International DMS Conference on Visual Languages and Sentient Systems (DMSVLSS)*.

[20] U. Deshpande and K. Keahey. 2017. Traffic-sensitive live migration of virtual machines. *Future Generation Computer Systems* 72 (Jul 2017), 118–128.

[21] S. Dhakal, M. M. Hayat, J. E. Pezoa, C. Yang, and D. A. Bader. 2007. Dynamic load balancing in distributed systems in the presence of delays: A regeneration-theory approach. *IEEE Transactions on Parallel and Distributed Systems* 18, 4 (Apr 2007), 485–497.

[22] S. Di, D. Kondo, and W. Cirne. 2012. Host load prediction in a Google compute cloud with a Bayesian model. In *Proc. IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. 1–11.

[23] D. Didona, F. Quaglia, P. Romano, and E. Torre. 2015. Enhancing performance prediction robustness by combining analytical modeling and machine learning. In *Proc. ACM/SPEC International Conference on Performance Engineering (ICPE)*. 145–156.

[24] H. T. Dinh, C. Lee, D. Niyato, and P. Wang. 2013. A survey of mobile cloud computing: architectures, applications, and approaches. *Wireless Communications and Mobile Computing* 13, 18 (Dec 2013), 1587–1611.

[25] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen. 2014. Multi-agent based architecture for dynamic VM consolidation in cloud data centers. In *Proc. EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*. 111–118.

[26] T. C. Ferreto, M. AS. Netto, R. N. Calheiros, and C. AF. De Rose. 2011. Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems* 27, 8 (Oct 2011), 1027–1034.

[27] M. Fowler and J. Lewis. 2014. Microservices – a definition of this new architectural term. (Mar 2014). Retrieved Mar 12, 2018 from http://martinfowler.com/articles/microservices.html

[28] M. H. Ghahramani, M. Zhou, and C. T. Hon. 2017. Toward cloud computing QoS architecture: analysis of cloud systems and cloud services. *IEEE/CAA Journal of Automatica Sinica* 4, 1 (Jan 2017), 6–18.

[29] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. 2011. Dominant resource fairness: Fair allocation of multiple resource types. In *Proc. USENIX Conference on Networked Systems Design and Implementation*. 323–336.

[30] Z. Gong, X. Gu, and J. Wilkes. 2010. PRESS: Predictive elastic resource scaling for cloud systems. In *Proc. IEEE International Conference on Network and Service Management (CNSM)*. 9–16.

[31] I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep learning*. The MIT Press, Cambridge.

[32] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella. 2014. Multi-resource packing for cluster schedulers. *ACM SIGCOMM Computer Communication Review* 44, 4 (Oct 2014), 455–466.

[33] A. Gulati, C. Kumar, and I. Ahmad. 2009. Storage workload characterization and consolidation in virtualized environments. In *Proc. International Workshop on Virtualization Performance: Analysis, Characterization, and Tools (VPACT)*.

[34] H. Guo and J. Liu. 2018. Collaborative computation offloading for multi-access edge computing over fiber-wireless networks. *IEEE Transactions on Vehicular Technology* 67, 5 (May 2018), 4514–4526.

[35] Y. Guo and W. Yao. 2018. A container scheduling strategy based on neighborhood division in micro service. In *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS)*. 1–6.

[36] L. Gupta, M. Samaka, R. Jain, A. Erbad, D. Bhamare, and H. A. Chan. 2017. Fault and performance management in multi-cloud based NFV using shallow and deep predictive structures. *Journal of Reliable Intelligent Environments* 3, 4 (Dec 2017), 221–231.

[37] L. Gupta, M. Samaka, R. Jain, A. Erbad, D. Bhamare, and C. Metz. 2017. COLAP: a predictive framework for service function chain placement in a multi-cloud environment. In *Proc. IEEE Annual Computing and Communication Workshop and Conference (CCWC)*. 1–9.

[38] T. Hastie, R. Tibshirani, and J. Friedman. 2001. *The elements of statistical learning: Data mining, inference, and prediction* (1 ed.). Springer, New York.

[39] N. Herbst, A. Amin, A. Andrzejak, L. Grunske, S. Kounev, O. J. Mengshoel, and P. Sundararajan. 2017. Online workload forecasting. In *Self-Aware Computing Systems*. Springer, Cham, 529–553.

[40] L. Hu, X.-L. Che, and S.-Q. Zheng. 2012. Online system for grid resource monitoring and machine learning-based prediction. *IEEE Transactions on Parallel and Distributed Systems* 23, 1 (Jan 2012), 134–145.

[41] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young. 2015. Mobile edge computing – A key technology towards 5G. ETSI White Paper. (Sep 2015). Retrieved Oct 2, 2018 from http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[42] Q. Huang, S. Su, S. Xu, J. Li, P. Xu, and K. Shuang. 2013. Migration-based elastic consolidation scheduling in cloud data center. In *Proc. IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*. 93–97.

[43] A. R. Hummaida, N. W. Paton, and R. Sakellariou. 2016. Adaptation in cloud resource configuration: a survey. *Journal of Cloud Computing* 5, 1 (Dec 2016).

[44] IBM. 2013. Smarter wireless networks. (Feb 2013). Retrieved Oct 2, 2018 from https://www.ibm.com/services/multimedia/Smarter_wireless_networks.pdf

[45] Z. Jia, J. Zhan, L. Wang, R. Han, S. A. McKee, Q. Yang, C. Luo, and J. Li. 2014. Characterizing and subsetting big data workloads. In *Proc. IEEE International Symposium on Workload Characterization (IISWC)*. 191–201.

[46] H. Kameda, E.-Z. Said Fathyt, I. Ryut, and J. Lis. 2000. A performance comparison of dynamic vs. static load balancing policies in a mainframe - Personal computer network model. In *Proc. IEEE Conference on Decision and Control*. 1415–1420.

[47] S. Kedar. 2017. Get Ready for the Holidays with Cloudlets. (Oct 2017). Retrieved Mar 12, 2018 from https://blogs.akamai.com/2017/08/get-ready-for-the-holidays-with-cloudlets.html

[48] A. Khan, X. Yan, S. Tao, and N. Anerousis. 2012. Workload characterization and prediction in the cloud: A multiple time series approach. In *Proc. IEEE Network Operations and Management Symposium (NOMS)*. 1287–1294.

[49] A. R. Khan, M. Othman, S. A. Madani, and S. U. Khan. 2014. A Survey of Mobile Cloud Computing Application Models. *IEEE Communications Surveys & Tutorials* 16, 1 (1^{st} quarter 2014), 393–413.

[50] K. Kim, C. Lee, J. H. Jung, and W. W. Ro. 2014. Workload synthesis: Generating benchmark workloads from statistical execution profile. In *Proc. IEEE International Symposium on Workload Characterization (IISWC)*. 120–129.

[51] F. Klinaku, M. Frank, and S. Becker. 2018. CAUS: An elasticity controller for a containerized microservice. In *Proc. ACM/SPEC International Conference on Performance Engineering (ICPE)*. 93–98.

[52] H.-J. Ku, J.-H. Jung, and G.-I. Kwon. 2017. A study on reinforcement learning based SFC path selection in SDN/NFV. *International Journal of Applied Engineering Research* 12, 12 (2017), 3439–3443.

[53] J. Kumar and A. K. Singh. 2016. Dynamic resource scaling in cloud using neural network and black hole algorithm. In *Proc. IEEE International Conference on Eco-friendly Computing and Communication Systems (ICECCS)*. 63–67.

[54] J. Kumar and A. K. Singh. 2018. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems* 81 (Apr 2018), 41–52.

[55] KVM. 2015. Migration – KVM. (2015). Retrieved Oct 2, 2018 from https://www.linux-kvm.org/index.php?title=Migration&oldid=173268

[56] T. Le Duc and P-O. Östberg. 2018. Application, workload, and infrastructure models for virtualized content delivery networks deployed in edge computing environments. In *Proc. IEEE International Conference on Computer Communication and Networks (ICCCN)*. 1–7.

[57] Y.-T. Lee and K.-T. Chen. 2010. Is server consolidation beneficial to MMORPG? A case study of World of Warcraft. In *Proc. IEEE International Conference on Cloud Computing (CLOUD)*. 435–442.

[58] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li. 2018. Edge-oriented computing paradigms: A survey on architecture design and system management. *ACM Computing Surveys (CSUR)* 51, 2 (Apr 2018).

[59] Y. Li, H. Hu, Y. Wen, and J. Zhang. 2016. Learning-based power prediction for data centre operations via deep neural networks. In *Proc. ACM International Workshop on Energy Efficient Data Centres (E2DC)*.

[60] B. Liu, Y. Lin, and Y. Chen. 2016. Quantitative workload analysis and prediction using Google cluster traces. In *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM)*. 935–940.

[61] C. Liu, C. Liu, Y. Shang, S. Chen, B. Cheng, and J. Chen. 2017. An adaptive prediction approach based on workload pattern discrimination in the cloud. *Journal of Network and Computer Applications* 80 (Feb 2017), 35–44.

[62] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. Foy, and Y. Zhang. 2018. Mobile edge cloud system: Architectures, challenges, and approaches. *IEEE Systems Journal* 12, 3 (Sep 2018), 2495–2508.

[63] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, and Y. Wang. 2017. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In *Proc. IEEE International Conference on Distributed Computing Systems (ICDCS)*. 372–382.

[64] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. H. Andrew. 2015. Greening geographical load balancing. *IEEE/ACM Transactions on Networking* 23, 2 (Apr 2015), 657–671.

[65] D. Magalhães, R. N. Calheiros, R. Buyya, and D. G. Gomes. 2015. Workload modeling for resource usage analysis and simulation in cloud computing. *Computers & Electrical Engineering* 47 (Oct 2015), 69–81.

[66] Y. Mansouri, A. Nadjaran Toosi, and R. Buyya. 2017. Cost optimization for dynamic replication and migration of data in cloud data centers. *IEEE Transactions on Cloud Computing (Early Access)* (Jan 2017), 1–1. https://doi.org/10.1109/TCC.2017.2659728

[67] R. Marcus and O. Papaemmanouil. 2016. Workload management for cloud databases via machine learning. In *Proc. IEEE International Conference on Data Engineering Workshops (ICDEW)*. 27–30.

[68] G. Mazlami, J. Cito, and P. Leitner. 2017. Extraction of microservices from monolithic software architectures. In *Proc. IEEE International Conference on Web Services (ICWS)*. 524–531.

[69] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett, et al. 2017. Knowledge-defined networking. *ACM SIGCOMM Computer Communication Review* 47, 3 (Sep 2017), 2–10.

[70] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan. 2010. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In *Proc. IEEE International Conference on Services Computing (SCC)*. 514–521.

[71] Microsoft. 2016. Hyper-V technology overview. (Nov 2016). Retrieved Oct 2, 2018 from https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview

[72] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba. 2017. Topology-aware prediction of virtual network function resource requirements. *IEEE Transactions on Network and Service Management* 14, 1 (Mar 2017), 106–120.

[73] A. S. Milani and N. J. Navimipour. 2016. Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends. *Journal of Network and Computer Applications* 71 (Aug 2016), 86–98.

[74] T. Miyazawa, V. P. Kafle, and H. Harai. 2017. Reinforcement learning based dynamic resource migration for virtual networks. In *Proc. IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 428–434.

[75] N. Mohan and J. Kangasharju. 2016. Edge-Fog cloud: A distributed cloud for Internet of Things computations. In *Proc. IEEE Cloudification of the Internet of Things (CIoT)*.

[76] L. R. Moore, K. Bean, and T. Ellahi. 2013. Transforming reactive auto-scaling into proactive auto-scaling. In *Proc. ACM International Workshop on Cloud Data and Platforms*. 7–12.

[77] D. Mukerjee, S. Dhara, S. C. Borst, and J. S.H. van Leeuwaarden. 2017. Optimal service elasticity in large-scale distributed systems. *ACM SIGMETRICS Performance Evaluation Review* 45, 1 (Jun 2017), 3–3.

[78] M. Mukherjee, L. Shu, and D. Wang. 2018. Survey of fog computing: Fundamental, network applications, and research challenges. *IEEE Communications Surveys & Tutorials* 20, 3 (3rd quarter 2018), 1826–1857.

[79] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung. 2017. The impact of database layer on auto-scaling decisions in a 3-tier web services cloud resource provisioning. In *Proc. IEEE Annual Computer Software and Applications Conference (COMPSAC)*. 401–406.

[80] Y. Niu, F. Liu, and Z. Li. 2018. Load Balancing Across Microservices. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*. 198–206.

[81] M. Noshy, A. Ibrahim, and H. A. Ali. 2018. Optimization of live virtual machine migration in cloud computing: A survey and future directions. *Journal of Network and Computer Applications* 110 (May 2018), 1–10.

[82] OpenStack. 2017. Documentation for Pike. (Aug 2017). Retrieved Oct 2, 2018 from https://docs.openstack.org/pike/

[83] Oracle. 2016. Oracle VM and Oracle Linux: Engineered for open cloud. (Aug 2016). Retrieved Oct 2, 2018 from http://www.oracle.com/us/technologies/virtualization/ovm-linux-engineered-for-open-cloud-3124867.pdf

[84] P-O. Östberg *et al.* 2017. Reliable Capacity Provisioning for Distributed Cloud/Edge/Fog Computing Application. In *Proc. IEEE European Conference on Networks and Communications (EuCNC)*.

[85] J. Pan and J. McElhannon. 2018. Future edge cloud and edge computing for Internet of Things applications. *IEEE Internet of Things Journal* 5, 1 (Feb 2018), 439–449.

[86] A. Paya and D. C. Marinescu. 2017. Energy-aware load balancing and application scaling for the cloud ecosystem. *IEEE Transactions on Cloud Computing* 5, 1 (Jan 2017), 15–27.

[87] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo. 2018. Optimus: An efficient dynamic resource scheduler for deep learning clusters. In *Proc. EuroSys Conference (EuroSys '18)*. 1–14.

[88] V. Persico, D. Grimaldi, A. Pescapè, A. Salvi, and S. Santini. 2017. A fuzzy approach based on heterogeneous metrics for scaling out public clouds. *IEEE Transactions on Parallel and Distributed Systems* 28, 8 (Aug 2017), 2117–2130.

[89] W. Rankothge, J. Ma, F. Le, A. Russo, and J. Lobo. 2015. Towards making network function virtualization a cloud computing service. In *Proc. IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 89–97.

[90] C. Reiss, J. Wilkes, and J. L. Hellerstein. 2011. Google cluster-usage traces: format + schema. *Google Inc., White Paper* (2011), 1–14.

[91] V. Riccobene, M. J. McGrath, M.-A. Kourtis, G. Xilouris, and H. Koumaras. 2016. Automated generation of VNF deployment rules using infrastructure affinity characterization. In *Proc. IEEE NetSoft Conference and Workshops (NetSoft)*. 226–233.

[92] C. Richardson. 2015. Introduction to Microservices. (May 2015). Retrieved Oct 2, 2018 from https://www.nginx.com/blog/introduction-to-microservices/

[93] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato. 2017. Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control. *IEEE Transactions on Computers (TC)* 66, 5 (May 2017), 810–819.

[94] R. Roman, J. Lopez, and M. Mambo. 2018. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems* 78, 1 (Jan 2018), 680–698.

[95] A. Roozbeh, J. Soares, G. Q. Maguire, F. Wuhib, C. Padala, M. Mahloo, D. Turull, V. Yadhav, and D. Kostić. 2018. Software-defined "hardware" infrastructures: A survey on enabling technologies and open research directions. *IEEE Communications Surveys & Tutorials* 20, 3 (3rd quarter 2018), 2454–2485.

[96] O. Runsewe and N. Samaan. 2017. Cloud resource scaling for big data streaming applications using a layered multi-dimensional hidden Markov model. In *Proc. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. 848–857.

[97] A. Sangpetch, O. Sangpetch, N. Juangmarisakul, and S. Warodom. 2017. Thoth: Automatic Resource Management with Machine Learning for Container-based Cloud Platform. In *Proc. International Conference on Cloud Computing and Services Science (CLOSER)*. 103–111.

[98] M. Satyanarayanan. 2012. Elijah: Cloudlet-based Edge Computing. (Jun 2012). Retrieved Mar 12, 2018 from http://elijah.cs.cmu.edu/

[99] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, and K. Ha. 2013. The role of cloudlets in hostile environments. *IEEE Pervasive Computing* 12, 4 (Oct 2013), 40–49.

[100] R. Shi, J. Zhang, W. Chu, Q. Bao, X. Jin, C. Gong, Q. Zhu, C. Yu, and S. Rosenberg. 2015. MDP and machine learning-based cost-optimization of dynamic resource allocation for network function virtualization. In *Proc. IEEE International Conference on Services Computing (SCC)*. 65–73.

[101] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison. 2017. The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective. *IEEE Journal on Selected Areas in Communications* 35, 11 (Oct 2017), 2586–2595.

[102] C. Sieber, A. Basta, A. Blenk, and W. Kellerer. 2016. Online resource mapping for SDN network hypervisors using machine learning. In *Proc. IEEE NetSoft Conference and Workshops (NetSoft)*. 78–82.

[103] C. Sieber, A. Obermair, and W. Kellerer. 2017. Online learning and adaptation of network hypervisor performance models. In *Proc. IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 1204–1212.

[104] A. Sill. 2016. The design and architecture of microservices. *IEEE Cloud Computing* 3, 5 (Sep-Oct 2016), 76–80.

[105] S. Singh and I. Chana. 2016. Cloud resource provisioning: survey, status and future research directions. *Springer Knowledge and Information Systems* 49, 3 (Dec 2016), 1005–1069.

[106] B. Speitkamp and M. Bichler. 2010. A mathematical programming approach for server consolidation problems in virtualized data centers. *IEEE Transactions on services computing* 3, 4 (Oct-Dec 2010), 266–278.

[107] J. Summers, T. Brecht, D. Eager, and A. Gutarin. 2016. Characterizing the workload of a Netflix streaming video server. In *Proc. IEEE International Symposium on Workload Characterization (IISWC)*. 1–12.

[108] G. Sun, D. Liao, V. Anand, D. Zhao, and H. Yu. 2016. A new technique for efficient live migration of multiple virtual machines. *Future Generation Computer Systems* 55 (Feb 2016), 74–86.

[109] Y. Sun, B. J. van Wyk, and Z. Wang. 2011. A New Multi-swarm Multi-objective Particle Swarm Optimization Based on Pareto Front Set. In *Proc. International Conference on Intelligent Computing*. Springer, 203–210.

[110] Z. Tang, X. Zhou, F. Zhang, W. Jia, and W. Zhao. 2018. Migration modeling and learning algorithms for containers in fog computing. *IEEE Transactions on Services Computing (Early Access)* (Apr 2018), 1–1. https://doi.org/10.1109/TSC.2018.2827070

[111] W. Tärneberg, A. Mehta, E. Wadbro, J. Tordsson, J. Eker, M. Kihl, and E. Elmroth. 2017. Dynamic application placement in the mobile cloud network. *Future Generation Computer Systems* 70 (May 2017), 163–177.

[112] N. T. Ti and L. B. Le. 2017. Computation offloading leveraging computing resources from edge cloud and mobile peers. In *Proc. IEEE International Conference on Communications (ICC)*. 1–6.

[113] M. Trivedi and H. Somani. 2016. A survey on resource provisioning using machine learning in cloud computing. *International Journal of Engineering Development and Research* 4, 4 (Nov 2016), 546–549.

[114] Carnegie Mellon University. 2012. OpenStack++. (2012). https://github.com/cmusatyalab/elijah-openstack

[115] A. Varasteh and M. Goudarzi. 2017. Server consolidation techniques in virtualized data centers: A survey. *IEEE Systems Journal* 11, 2 (Jun 2017), 772–783.

[116] VMware. 2017. VMware - vSphere vMotion. (Jan 2017). Retrieved Mar 12, 2018 from https://www.vmware.com/products/vsphere/vmotion.html

[117] S. Wang, M. Zafer, and K. K. Leung. 2017. Online placement of multi-component applications in edge computing environments. *IEEE Access* 5 (Feb 2017), 2514–2533.

[118] P. Watson and M. Little. 2014. Multilevel security for deploying distributed applications on clouds, devices and things. In *Proc. IEEE International Conference on Cloud Computing Technology and Science*. 380–385.

[119] R. Wolski and J. Brevik. 2014. Using parametric models to represent private cloud workloads. *IEEE Transactions on Services Computing* 7, 4 (Oct-Dec 2014), 714–725.

[120] H. Wu, Y. Sun, and K. Wolter. 2018. Energy-efficient decision making for mobile cloud offloading. *IEEE Transactions on Cloud Computing (Early Access)* (Jan 2018), 1–1. https://doi.org/10.1109/TCC.2018.2789446

[121] M. Xu, A. V. Dastjerdi, and R. Buyya. 2016. Energy efficient scheduling of cloud application components with brownout. *IEEE Transactions on Sustainable Computing* 1, 2 (Jul-Dec 2016), 40–53.

[122] Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny. 2014. Characterizing Facebook's memcached workload. *IEEE Internet Computing* 18, 2 (Mar 2014), 41–49.

[123] J. Xue, F. Yan, R. Birke, L. Y. Chen, T. Scherer, and E. Smirni. 2015. PRACTISE: Robust prediction of data center time series. In *Proc. IEEE International Conference on Network and Service Management (CNSM)*. 126–134.

[124] R. Yang, X. Ouyang, Y. Chen, P. Townend, and J. Xu. 2018. Intelligent resource scheduling at scale: A machine learning perspective. In *Proc. IEEE Symposium on Service-Oriented System Engineering (SOSE)*. 132–141.

[125] S. A. Yousif and A. Al-Dulaimy. 2017. Clustering cloud workload traces to improve the performance of cloud data centers. In *Proc. World Congress on Engineering (WCE)*, Vol. 1.

[126] Y. Yu, V. Jindal, I-L. Yen, and F. Bastani. 2016. Integrating clustering and learning for improved workload prediction in the cloud. In *Proc. IEEE International Conference on Cloud Computing (CLOUD)*. 876–879.

[127] F. Zhang, X. Fu, and R. Yahyapour. 2016. LayerMover: Storage migration of virtual machine across data centers based on three-layer image structure. In *Proc. IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 400–405.

[128] F. Zhang, X. Fu, and R. Yahyapour. 2017. CBase: A new paradigm for fast virtual machine migration across data centers. In *Proc. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 284–293.

[129] Q. Zhang, L. Cheng, and R. Boutaba. 2010. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications* 1, 1 (May 2010), 7–18.

[130] W. Zhang, P. Duan, L. T. Yang, F. Xia, Z. Li, Q. Lu, W. Gong, and S. Yang. 2017. Resource requests prediction in the cloud computing environment with a deep belief network. *Software: Practice and Experience* 47, 3 (Mar 2017), 473–488.

Received October 2018; revised March 2019; accepted June 2019