A Review on Computational Intelligence Techniques in Cloud and Edge Computing

Muhammad Asim, Yong Wang, *Senior Member, IEEE*, Kezhi Wang, *Senior Member, IEEE*, and Pei-Qiu Huang

***Abstract*—Cloud computing (CC) is a centralized computing paradigm that accumulates resources centrally and provides these resources to users through Internet. Although CC holds a large number of resources, it may not be acceptable by real-time mobile applications, as it is usually far away from users geographically. On the other hand, edge computing (EC), which distributes resources to the network edge, enjoys increasing popularity in the applications with low-latency and high-reliability requirements. EC provides resources in a decentralized manner, which can respond to users' requirements faster than the normal CC, but with limited computing capacities. As both CC and EC are resource-sensitive, several big issues arise, such as how to conduct job scheduling, resource allocation, and task offloading, which significantly influence the performance of the whole system. To tackle these issues, many optimization problems have been formulated. These optimization problems usually have complex properties, such as non-convexity and NP-hardness, which may not be addressed by the traditional convex optimization-based solutions. Computational intelligence (CI), consisting of a set of nature-inspired computational approaches, recently exhibits great potential in addressing these optimization problems in CC and EC. This article provides an overview of research problems in CC and EC and recent progresses in addressing them with the help of CI techniques. Informative discussions and future research trends are also presented, with the aim of offering insights to the readers and motivating new research directions.**

***Index Terms*—Cloud computing (CC), edge computing (EC), computational intelligence, evolutionary algorithms, swarm intelligence algorithms, fuzzy system, learning based system.**

List of Abbreviations

ACO	Ant Colony Optimization
AI	Artificial Intelligence
AIS	Artificial Immune System
BA	Bee Algorithm
BFA	Bacterial Foraging Algorithm
CC	Cloud Computing
CI	Computational Intelligence
C-RAN	Cloud-Radio Access Network
CSA	Cuckoo Search Algorithm
DE	Differential Evolution
EAs	Evolutionary Algorithms
EC	Edge Computing
FA	Firefly Algorithm
FC	Fog Computing
FI	Fuzzy Inference
FL	Fuzzy Logic
FS	Fuzzy System
GA	Genetic Algorithm
HS	Hybrid System
IaaS	Infrastructure-as-a-Service
IoT	Internet of Things
LBS	Learning Based System
MCC	Mobile Cloud Computing
MDP	Markov Decision Process
MEC	Mobile Edge Computing
NN	Neural Networks
PSO	Particle Swarm Optimization
QoS	Quality of Service
SIAs	Swarm Intelligence Algorithms

Manuscript received January 21, 2020; revised June 10, 2020; accepted July 2, 2020. Date of publication August 20, 2020; date of current version November 21, 2020. This work was supported in part by the National Natural Science Foundation of China under Grants 61673397 and 61976225, in part by the Beijing Advanced Innovation Center for Intelligent Robots and Systems under Grant 2018IRS06, and in part by the Foundational Research Funds for the Central Universities of Central South University under Grant 2020zzts521. *(Corresponding authors: Yong Wang; Kezhi Wang.)*

Muhammad Asim is with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: asimpk@csu.edu.cn).

Yong Wang and Pei-Qiu Huang are with the School of Automation, Central South University, Changsha 410083, China (e-mail: ywang@csu.edu.cn; pqhuang@csu.edu.cn).

Kezhi Wang is with the Department of Computer and Information Sciences, Northumbria University, Newcastle NE1 8ST, UK (e-mail: kezhi.wang@northumbria.ac.uk).

Digital Object Identifier 10.1109/TETCI.2020.3007905

I. Introduction

CLOUD computing (CC) is a paradigm of computing technologies that provides on demand services to its clients. The vision of CC is to offer computing, storage, and network resources centrally in the remote clouds, which is related to data centers, backhaul networks, and core networks [1], [2]. It is an architecture for allowing appropriate, pervasive, and on request access to a shared pool of configurable resources [3]. The large number of resources available in the central cloud can then be leveraged to deliver elastic computing capacity and storage to support resource-constrained devices. It has been driving the rapid growth of many Internet companies [4]. For example, the cloud business has risen to be the most profitable sector for Amazon [5], and Dropbox's success depends highly on the cloud service of Amazon. It can intensify collaboration, scalability, nimbleness, and availability for enterprises as well as users. CC offers services on a pay-as-you-go basis and reduces hardware and software costs,

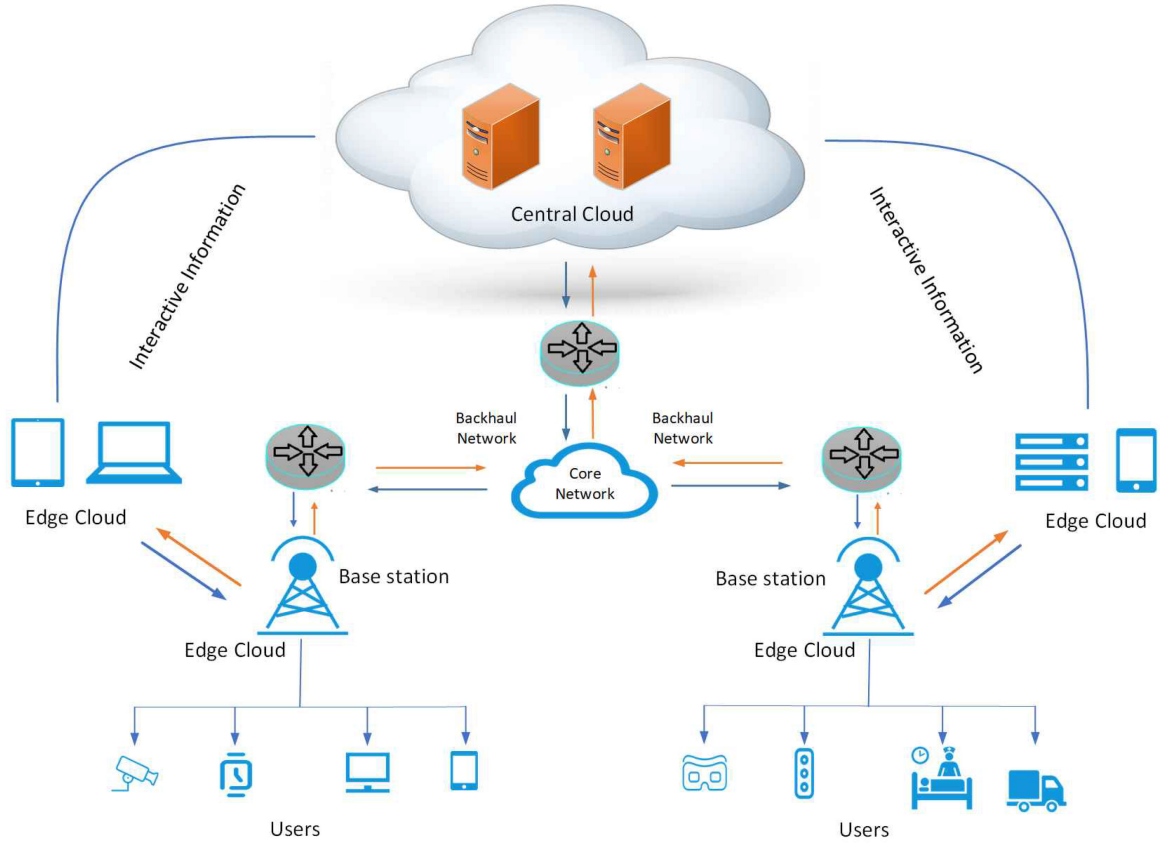


Fig. 1. Architecture of CC and EC.

energy, and carbon footprints by providing an optimized and sufficient computing environment [6]. In CC, users first offload tasks to the central cloud, and then the central cloud executes the tasks on behalf of users and returns results to users [7]. Although CC provides a vast number of resources, easy back and recovery, high accessibility as well as an eco-friendly environment for users (i.e., can be accessed anytime from anywhere on demand), it may not be capable of fulfilling the requirements of real-time applications with low latency and high reliability as the central cloud is far away from users.

On the other hand, edge computing (EC), which deploys the resources, data, and services from the central cloud to the network edge, enjoys increasing popularity recently. It enables analytics and knowledge generation to occur at or close to users' devices. In EC, users first offload tasks to the nearby edge cloud, and then the edge cloud executes their tasks and sends outcomes back to users. Fig. 1 presents a typical architecture of CC and EC, where edge clouds can provide computing and network resources between users and the central cloud. Compared with CC, EC can provide services with low latency, high security, and high mobility but with limited computing capacities. CC and EC facilitate a wide range of applications including virtual reality/augmented reality, online game, online video, etc.

In CC and EC, several metrics are related to the quality of service (QoS), such as security and privacy, latency, resource utilization, cost, energy consumption, throughput, and makespan. In order to improve these metrics, we need to consider several issues, such as job scheduling, resource allocation, and task offloading. For example, security and privacy can be improved by designing efficient task offloading schemes [8]. In addition, latency and energy consumption can be optimized by designing proper resource allocation [9] and task offloading schemes [10], [11]. Similarly, other metrics can be improved by considering job scheduling, resource allocation, and task offloading. To this end, many optimization problems have been formulated for tackling these issues. However, compared with optimization problems in other areas, these optimization problems are normally highly complex and NP-hard, which may include mixed/strongly-coupled variables, nonlinear constraints, multiple objectives, and bilevel structures. Therefore, they may not be solved by the traditional convex optimization-based methods. In addition, their optimal solutions should be found within a reasonable amount of time.

As a class of nature-inspired computational approaches, computational intelligence (CI) exhibits great potential in addressing complex optimization problems, which has attracted much attention from both academia and industry. CI includes evolutionary algorithms (EAs), swarm intelligence algorithms (SIAs), fuzzy system (FS), learning based system (LBS), and hybrid system (HS). These CI techniques have the capability to process imprecise information and search for approximate yet good enough solutions while ensuring robustness and computational tractability [12].

The aim of this paper is to present a review of CI techniques to address issues in CC and EC. Several survey papers and books have been devoted to CC and EC. For example, Hoang *et al.* [13] presented a survey on architecture, applications, and approaches in mobile CC (MCC). A short review on task scheduling in CC is carried out in [14]. QoS in CC is surveyed in [15] and [16]. Wu [17] studied multi-objective decision making for offloading decision in MCC. An extensive survey on MCC is given in [18]. A survey on challenges and opportunities in CC is presented in [19]. Zhou *et al.* [20] published a survey on artificial intelligence (AI) in EC. In [21], visions and challenges of EC are discussed. A survey on EC for Internet of things (IoT) is investigated in [22]. Liu *et al.* [23] surveyed systems and tools in EC. Peng *et al.* [24] studied service adaptation and provision in mobile EC (MEC). Mao *et al.* [4] elaborated on the communication perspective of MEC. Lin *et al.* presented a survey on computation offloading in EC. Wang *et al.* [25] focused on offloading algorithms, issues, methods, and perspectives in edge cloud. Moreover, some other surveys devoted to MEC are [26]–[31]. However, none of the aforementioned surveys considers CI techniques. In addition, some survey papers review CI techniques in CC or EC. For example, evolutionary approaches for resource management in CC are reviewed in [32] and [33]. Chopra and Bedi [34] reviewed the applications of fuzzy logic (FL) in CC. Deep learning (DL) in EC is studied in [35]. It is worth noting that the above-mentioned surveys only consider one particular type of CI for CC or EC.

This survey attempts to give a detailed review of the state-of-the-art CI techniques in CC and EC. Our paper is an ambitious effort to capture the interplay among CI, CC, and EC, instead of delving into one particular CI technique in CC and EC exclusively. The motivation behind this survey is to provide researchers with a glance of mutual relationship between CI and both CC and EC at a higher level.

The works introduced in this survey are taken from relevant journals, workshops, conference proceedings, and theses. This survey focuses on reviewing issues in CC and EC tackled by CI techniques, rather than a detailed study of CC/EC/CI techniques.

The rest of this paper is organized as follows. Section II presents an overview of the concepts related to CC and EC along with important metrics and critical issues. Section III introduces CI techniques used for addressing critical issues in CC and EC. Section IV reviews existing applications of CI techniques in CC and EC. Finally, Section V offers a detailed discussion on the use of CI techniques in CC and EC along with future research trends, followed by the conclusion in Section VI. For clarity, the organization of this paper is given in Fig. 2.

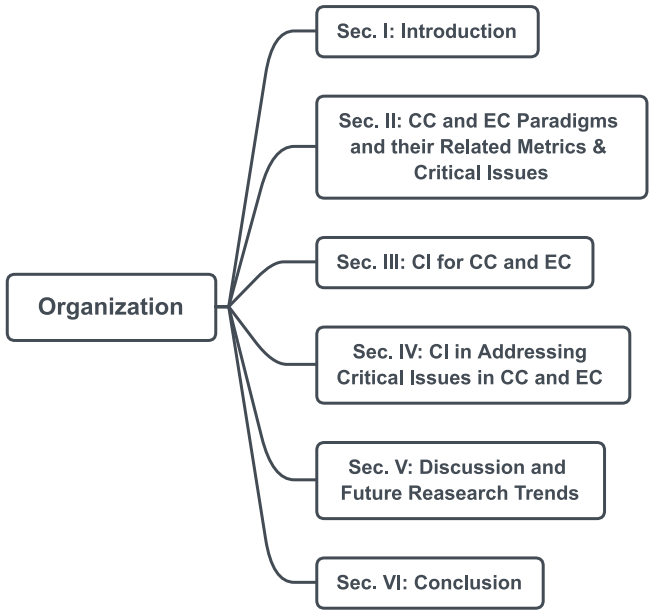


Fig. 2. Paper Organization.

II. CC and EC Paradigms and Their Related Metrics and Critical Issues

A. CC and EC Paradigms

CC and EC are complementary fields and their features are summarized in Table I.

TABLE I
Features of CC and EC

Feature	CC	EC
Latency	High	Low
Mobility	No	Yes
Architecture	Centralized	Distributed
Location Awareness	No	Yes
Security	Less Secure	More Secure
Service Access	Through Core	At Users' Devices
Availability	High	High
Geographic Distribution	No	Yes
Scalability	Medium	High
Reliability	Low	High
No. of devices connected	In Millions	In Billions
Resources	Huge	Limited

1) CC: In CC, different types of paradigms have been designed. Next, we introduce two typical paradigms of CC: MCC and cloud-radio access network (C-RAN).

- MCC is proposed to enrich resources for users. It merges CC, mobile computing, and wireless networks to enable service providers to help users to conduct computation-intensive tasks [18]. It provides several benefits to users like extending battery lifetime, improving storage capacity and processing power, and providing computation-intensive applications [13].
- C-RAN provides centralized processing, collaborative radio, and energy efficient infrastructure for RAN [36]. In this architecture, network computation-based tasks are performed in the central baseband unit, and the radio signals from distributed antennas are gathered through remote radio heads and transmitted to the cloud by optical transmission network. It offers better services without affecting coverage of network by reducing the number of cell sites, capital expenditures, and operating costs.

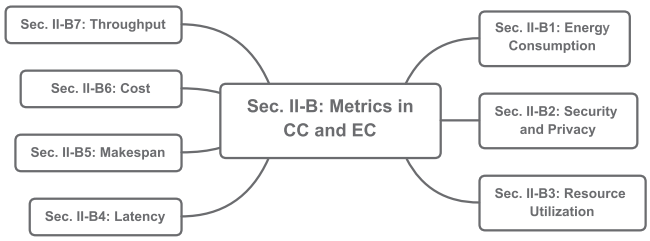


Fig. 3. Metrics in CC and EC.

2) EC: EC includes many paradigms, e.g., cloudlet, fog computing (FC), and MEC, which are discussed below.

- Cloudlet, introduced by Satyanarayanan *et al.* [37], [38], is a decentralized and widely distributed infrastructure, i.e., providing storage and processing close to users. It is also referred to as a "data center in a box". It is indeed proposed to fulfill the demands of users to reduce response time for new latency-sensitive applications without going to the main central cloud [39]. Although cloudlet has many benefits due to being close to users, it does not have enough computing capacity as the central cloud; thus, it is limited in providing computation-intensive services.
- FC was first discussed by Cisco in 2012 [40] and is referred to as an extension of services from the central cloud to the network edge. It reduces the amount of data sent to the central cloud for processing and storage [41]. It is a paradigm that furnishes computing, storage, and networking services close to users. FC provides better support for real-time applications, dense geographical distribution, low latency, and location awareness.
- The architecture of MEC was proposed by European Telecommunications Standards Institute in 2014, enabling the provision of resources close to users via RAN [42]. It can provide services with low latency and high bandwidth at the edge of radio network. In addition, MEC makes it possible to measure and improve network performance by setting up services like software defined network and network function virtualization [43].

Although EC can effectively overcome some problems of network congestion and long latency in CC, it has limited computation and communication capabilities [44]. On the other hand, although CC has rich computing resources, it may suffer from high latency. To address the above-mentioned problems, researchers have made some attempts to investigate the hierarchical computing (i.e., the collaboration between CC and EC), such as [44]–[46], where the tasks of users can be partially or jointly processed at both edge and central clouds.

B. Metrics

Over the past two decades, many researchers focused on diverse aspects of CC and EC, including security and privacy, energy consumption, resource utilization, etc. Some important metrics are given in Fig. 3 and discussed below.

1) Energy Consumption: EC may suffer from battery life problem. Even though CC is normally connected to the power grid, energy efficiency is still very important as it is relevant to the profit and economy of CC operators. The energy spent on end devices during task execution is a primary aspect to be considered. Total energy consumption includes static energy consumption and dynamic energy consumption [47]. Energy consumed by the system without considering any workload is called the static energy, while the dynamic energy is the energy consumed on the current cloud/edge resources by virtual machines (VMs). Usually, an edge cloud can be a small device like a laptop, and applies the battery. However, CC normally connects to power grid, and has much more power consumption than EC. Therefore, saving energy in EC is to keep edge cloud alive, whereas in CC, saving energy contributes to green society. For users, due to the long distance transmission, CC usually needs more energy than EC for offloading tasks.

2) Security and Privacy: Security and privacy is one of the important metrics to be considered for providing secure and trusted services to users. Many hackers try to interrupt or steal data while users offload and process their tasks at central/edge clouds. Security and privacy is a set of control-based technologies and policies designed to protect sensitive information, data, applications, and infrastructures [48]. CC is more vulnerable to be attacked by hackers due to the long distance transmission between users and the central cloud. In contrast, EC is a decentralized architecture, which is more secure than CC. Locally sharing, storing, exchanging, and analyzing data among edge clouds make it harder for hackers to get access to data. Moreover, real-time processing and response of EC make it difficult for malicious attackers to detect sensitive information of users. However, EC still inherits many security problems such as privacy leakage, forgery, tampering, spam, jamming, and impersonation [49].

3) Resource Utilization: If CC and EC infrastructures place VMs for resource utilization without any specific scheme, some resources may run out while others are idle. Thus, an efficient resource utilization scheme becomes important to get the maximum profit by utilizing resources properly [50]. Moreover, load balancing also needs to be considered for better performance. It is essential to distribute workloads across multiple nodes of the system to maintain stability, minimize latency, and improve resource utilization ratio in CC and EC. In CC, proper utilization of resources can save energy, host more users, and make more profits, whereas in EC, it can meet high-reliability and low-latency requirements.

4) Latency: A mass of novel mobile applications is emerging, most of them are latency-sensitive. Typically, latency is defined as the delay between a user's request and response of a service provider [51]. It has a high effect on the usability and enjoyability of end devices. CC is a large system and hosts a huge number of users, thus leading to routing problems such as VMs' connection problem. In addition, the long physical distance between users and the central cloud also leads to high latency. In contrast, edge clouds are placed at the network edge

and thus have a better capability to perform latency-sensitive tasks than CC.

5) Makespan: Makespan is the maximum required time to complete all assigned tasks [33], [52], including response time, execution time, waiting time, etc. It can be expressed as:

$$M = \max\{T_i\}, \quad i = 1, 2, \ldots, n \tag{1}$$

where T_i is the completion time of task i, and n is the number of tasks.

Compared with EC, there are a huge number of tasks needed to be processed in CC. Therefore, many researchers focused on makespan in CC only.

6) Cost: Since users and service providers usually belong to different entities, users have to pay for specific resources [51] such as computing or communication resources. The cost can consist of computing cost, running cost, and setup cost. The purpose of this metric is to reduce the cost of services provided to users. One of the most significant ways is to generate a program for services, which can manage the changing behavior of users and optimize costs for infrastructure maintenance and order.

7) Throughput: Throughput measures the rate at which data is successfully transferred between two endpoints. It concerns the ability of a service provider to handle the demands of users as it cannot directly track users' experience [51]. It is affected by latency and limitations existing at users' devices and central/edge clouds. Usually, EC has a higher throughput compared with CC, as it takes less time in responding to users' requests or transferring data to users.

C. Issues in CC and EC

In order to improve the above-mentioned metrics in CC and EC, we need to address the following issues.

1) Job Scheduling: The rising demands for services in CC and EC may lead to the imbalanced resource usage and drastically affect the performance of service providers; hence, job scheduling is a necessary prerequisite for improving QoS in CC and EC. The main aim of job scheduling is to order tasks in a specific way. There are two types of job scheduling: static and dynamic scheduling. In the static scheduling, all jobs arrive at the same time and are assigned to VMs in a static way, while in the dynamic scheduling, all jobs once arrive, they are scheduled instantly.

2) Resource Allocation: Improper distributions of resources in central/edge clouds lead to performance degradation of applications. It is becoming more and more difficult to allocate resources accurately to meet the demands of users and the service level agreements. The main aim of resource allocation is to assign available resources to the demanded cloud/edge applications over the Internet or wireless networks. Resource allocation should be provided to both users and service providers and, as a result, users can access good-quality services through cloud/edge service providers.

3) Task Offloading: The rapid growth of Internet services has yielded a variety of computation-intensive applications such as virtual reality applications. If applications are executed in the end devices, it leads to high computing costs, while if they are executed in central/edge clouds, it may take high transmission costs. Thus, users have to decide whether to offload their applications to central/edge clouds or not. Task offloading is the process that maps users' tasks to suitable resources in the form of VMs to execute. Usually, in CC, users only need to consider whether to offload their tasks or not as there is only one data center. But in EC, users have to further consider where to offload their tasks as there are many edge clouds. Task offloading is a critical issue to improve QoS in CC and EC.

4) Joint Issues: A growing number of applications are multi-sensitive. For instance, some tasks are both resource-intensive and delay-sensitive, e.g., face recognition [53] and augmented reality applications [54]. Researchers started addressing multiple issues together in CC and EC, which are termed as joint issues.

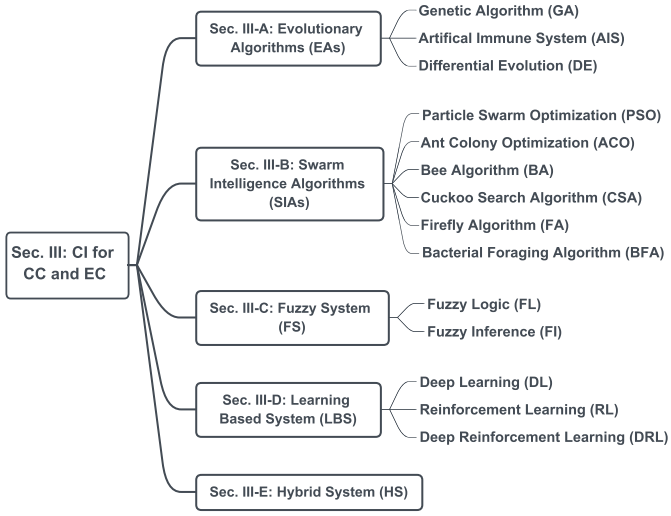


Fig. 4. Main paradigms of CI family used in CC and EC.

III. CI FOR CC AND EC

CI is a developed paradigm of intelligent systems, which is a set of nature-inspired computational methodologies and approaches to address complex real-world problems. It has attracted much attention from researchers and practitioners around the world. We focus on five categories of CI techniques applied in CC and EC, i.e., EAs, SIAs, FS, LBS, and HS, which are given in Fig. 4.

A. EAs

EAs try to mimic the process of natural evolution to find suitable solutions to optimization problems [55]. EAs follow different evolutionary mechanisms, however the basic ideas of these mechanisms are the same. Some commonly used EAs for CC and EC are given below.

- Genetic algorithm (GA) was proposed by Holland in 1975 inspired by Darwin's principle of survival of the fittest [56]. GA operates on a set of individuals called a population. Genetic operators, i.e., crossover and mutation, are applied to produce offspring from the parent population. Then, better

individuals are selected for the next generation through a selection operator based on the fitness values. This process continues until the stopping criterion is met. Finally, the best individual is returned as the best solution found for an optimization problem. GA has been widely applied to address issues in CC and EC, such as [57]–[61].

- Artificial immune system (AIS) is a kind of intelligent computational model inspired by the principles of human immune system with the characteristics of self organization, learning, memory, adaptation, robustness, and scalability [62]. In the initialization phase, candidate antibodies are randomly produced to form a population. The affinity of each antibody is evaluated by the fitness function. In each iteration, each antibody is cloned to a number of offspring. Then, these offspring experience the mutation operator. Only the one with the highest affinity is selected. Unlike GA, there is no crossover operator in AIS. AIS has been used for addressing issues in CC and EC, such as [63]–[65].
- Differential evolution (DE) was developed by Storn and Price in 1997 [66]. Due to the simple structure, ease of implementation, and robustness, it has been successfully applied to complex optimization problems, such as resource allocation [67] and job scheduling [68] in CC and EC. In DE, the first step is to initialize a population randomly. After initialization, a mutant vector is generated by adding the weighted difference vector of individuals to another individual using a mutation operator. Then, a crossover operator is implemented to obtain a trial vector. Afterward, a selection operator is applied to select better individuals for the next generation. Finally, the best individual is returned as the solution of an optimization problem.

B. SIAs

SIAs are a type of nature-inspired algorithms based on interaction among living organisms [69]. They can be described by the collective behaviors of living organisms working under specific rules. Some commonly used SIAs in CC and EC are discussed in the following:

- Particle swarm optimization (PSO) was proposed by Kennedy and Eberhart [70] in 1995. It is inspired by the social behaviors of the groups of population in nature such as animal herds, bird flocking, or schooling of fish. In PSO, a population of particles is first generated. In each iteration, each particle has an adaptable velocity, according to which it moves in the search space to update position and velocity. Moreover, each particle has a memory, remembering the best position it has ever visited. Thus, its movement is an aggregated acceleration toward its best previously visited position and the best position ever detected/visited by the swarm [71]. This process continues until the stopping criterion is met. PSO has been used to tackle issues in CC [72]–[74].
- Ant colony optimization (ACO), introduced in the early 1990s by Dorigo *et al.* [75], [76], is a technique for optimization. The inspiring source of ACO is the foraging behavior of real ant colonies [77]. In ACO, firstly a finite set of solution components is derived. Secondly, one has to define a set of pheromone values called the pheromone model, which is one of the central components of ACO. In each iteration, ACO assigns higher pheromone values to good solution components probabilistically. ACO has been applied to address job scheduling and resource allocation in CC and EC [78]–[81].
- Bee algorithm (BA), introduced by Pham [82] in 2005, is inspired by the natural foraging behavior of honey bees. In BA, the agent population is divided into a small set of scouts and a large set of foragers. The scouts randomly sample in the solution space, and evaluate the visited flower patches (i.e., locations). In each iteration, BA compares the new solutions discovered by the scouts with the best-so-far solution. The solutions are ordered according to their fitness values, and the highest ranking ones are selected for local search. BA performs exploitative neighborhood search and random explorative search. BA has been applied to job scheduling in CC and EC [83]–[85].
- Cuckoo search algorithm (CSA) is inspired by the living behaviour of a kind of bird named cuckoo and was developed by Yang and Deb in 2009 [86]. In CSA, the adult cuckoos lay eggs in the habitat of other birds. If these eggs are not found and thrown by host birds, they grow and become mature cuckoos. CSA begins with a primary population i.e., a group of cuckoos. In each iteration, CSA performs the following steps. First, individuals are generated by making modifications in existing individuals. After that, the new individuals are evaluated by the fitness function. The new individuals are compared with existing individuals, if they are better than existing individuals, they replace existing individuals in the population. Job scheduling in CC and EC has been addressed by using CSA [87], [88].
- Firefly algorithm (FA) was developed by Yang [89] in 2008. It is inspired by the flashing pattern and behavior of fireflies. In FA, a population of fireflies is generated randomly. In each iteration, one firefly can be fascinated by other fireflies regardless of its sex. The less brighter firefly is fascinated by the brighter one in the population. The brightness of a firefly is set on the landscape of the fitness function [90]. In case there is no brighter one, the fireflies move randomly and form a new population for the next iteration. The light intensity of fireflies is updated by evaluating the new population. Resource allocation and job scheduling in CC and EC have been tackled by using FA [91]–[93].
- Bacterial foraging algorithm (BFA) is a swarm optimization algorithm inspired by colonies of escherichia coli and proposed by Passino [94]. In BFA, individuals are first generated randomly. Then, for all individuals, the chemotaxis process is carried out. The process in which an organism moves with the gradient of a substance concentration is called chemotaxis. After the chemotaxis process, individuals are reproduced and dispersed. Elimination-dispersion is applied as the final step, in which many random individuals

are eliminated and many new individuals are generated. BFA has been used for job scheduling in CC [95], [96].

C. FS

FS is a type of CI family that resembles human reasoning. FS is a unique system that can handle numerical data and linguistic knowledge at the same time. We have found two kinds of FS for CC and EC, which are described below.

- FL was initiated by Zadeh in 1965 [97] during the development of theory of fuzzy sets. It works on approximate reasoning rather than exact approximation. The main difference between FL and traditional logic is that FL represents natural language and human thinking. In traditional logic, each member of a binary set has a logical value either 0 or 1, which shows that either it fully belongs to the set or not. There is no concept of partial membership in the traditional set theory [98]. Fuzzy modeling provides the best alternative for fuzziness and obscurities. FL has been applied to address issues in CC and EC, for instance, job scheduling [99]–[101], resource allocation [100], [102], and task offloading [103].
- Fuzzy inference (FI) is a model that can infer a crisp value as an input from a set of inputs, their fuzzy set, membership function, and a set of inference rules [104]. FI is usually constructed as follows. Firstly, inputs for the model are fuzzified through a membership function linked with the predefined fuzzy set. Linguistics terms are used to label the fuzzy set. Afterward, the firing strength is calculated for each rule, showing that some rules are more important than others in approaching the conclusion. All firing strengths are then aggregated and weighted for all rules, thus producing a fuzzy value. Finally, this fuzzified value is defuzzified by using a proper method. FI has been used for resource allocation [105] and task offloading [106], [107] in CC and EC.

D. LBS

LBS is inspired by the learning behaviour of living organisms. It uncovers the relationship between a set of nominal features and a set of states or objects [119]. In the following, we present the three most prominent kinds of LBS.

- DL is inspired by human's thinking ability, i.e., the mechanisms of human brain and neurons for processing signals, and was proposed by Hinton *et al.* [120] in 2006. It can cataract layers to extract features from the input data and eventually form a decision [121]. DL consists of input layer, output layer, and one or more hidden layers. In the input layer, the state of environment is represented in a suitable numerical form, which can be used as inputs of the network. Afterward, an activation function (e.g., Sigmoid function, Tanh function, or rectified linear unit function) is applied to represent or interrupt the recollection of results. Subsequently, all weights among layers are updated. Finally, the structure of DL is constructed. DL has been used for task offloading in EC [8], [122], [123].
- Reinforcement learning (RL) [124], [125] determines an optimal policy dictating which actions to take at certain states to achieve the highest possible reward. The problem such as resource allocation [126] in CC and EC is often formulated as a Markov decision process (MDP), where there is a set of states and actions. In RL, an agent under a given state selects a suitable action from the set of actions, receives a reward, and moves into a new state where it chooses another action from the updated set of actions. This process continues until a specific stopping criterion is met. RL has been applied to address issues in CC and EC [126]–[131].
- Deep RL (DRL) can be seen as a class of new efficient learning algorithm by combining DL with RL [132]–[134]. It is a powerful model that implements DL architectures such as deep neural networks with RL algorithms like Q-learning to scale and solve problems in different areas [135]. It has been effectively used for addressing issues in CC and EC, for instance, job scheduling [136], [137], resource allocation [138], task offloading [139]–[142], and joint issues [143]–[145]. Two commonly used DRL algorithms are deep Q-learning and deep Q-network.

E. HS

HS is a system combining two or more CI techniques to strengthen their individual capability. For example, genetic-fuzzy system is proposed for automating maritime risk assessment [146], which is a hybrid system of GA and FL. Also, EAs and SIAs are often combined as both have similar behaviors. Similarly, fuzzy adaptive theory [147] is sometimes hybridized with FL and neural networks (NN) [12]. Recently, some hybrid algorithms have been applied to address critical issues in CC [148]–[151].

IV. CI in Addressing Critical Issues in CC and EC

This section reviews the CI techniques introduced in Section III to solve the critical issues in CC and EC: job scheduling, resource allocation, task offloading, and joint issues.

A. EAs in CC and EC

This subsection is relevant to the works on the use of EAs in CC and EC, which are summarized in Table II.

1) EAs for Job Scheduling: Hu *et al.* [57] proposed an improved adaptive GA based on a priority mechanism for task scheduling in CC. This algorithm ensures the least execution time for job scheduling and guarantees the QoS requirements of users. It outperforms an adaptive GA and some other GAs in terms of convergence speed, feasibility and effectiveness. Agarwal and Srivastava [58] proposed a GA for task scheduling in CC, which distributes the loads among VMs effectively to optimize the overall response time. Experimental results show that the proposed algorithm outperforms some existing techniques such as greedy-based techniques and First Come First Serve in terms of overall response time. Liu *et al.* [59] developed a job scheduling model in CC based on multi-objective GA, which

TABLE II
EAs FOR ADDRESSING ISSUES IN CC AND EC

Issue	EA	Metric	Paradigm	Reference
Job Scheduling	GA	execution time	CC	[57]
	GA	response time	CC	[58]
	GA	energy consumption, cost	CC	[59]
	GA	makespan	CC	[108]
	GA	makespan, resource utilization	CC	[109]
	GA	makespan, computing cost	CC	[110]
	GA	execution time, cost	FC	[61]
	AIS	response time	FC	[111]
Resource Allocation	DE	resource utilization, makespan	CC	[67]
	GA	makespan	CC	[112]
	GA	energy consumption, latency	MEC	[9]
Task Offloading	AIS	security and privacy	CC	[63]
	GA	execution time, energy consumption	MCC	[113]
	GA	energy consumption	MCC	[114]
	GA	latency	FC	[60]
	AIS	security and privacy	EC	[64]
	AIS	security and privacy	FC	[65]
	GA	latency	EC	[10]
	GA	energy consumption	MEC	[11]
	GA	execution time, energy consumption	MCC	[115]
	GA	energy consumption, latency	MEC	[116]
Joint issues	DE	energy consumption	MEC	[68]
	GA	energy consumption	MEC	[117]
	GA	makespan	MEC	[118]

aims to minimize the energy consumption and maximize the profit of service. The proposed model has several components to analyze the applications and to allocate the appropriate resources to the applications. Experimental results show that the proposed model can obtain a higher profit, while consuming lower energy. Zarina *et al.* [108] investigated GA-based optimal job scheduling and load balancing in CC. They combined a GA with three traditional scheduling techniques: min-min, max-min and suffrage. In the first phase, these three traditional scheduling techniques are applied to obtain the minimum completion time for a given job at each VM. After that, GA is applied to attain better QoS by utilizing available resources. Experimental results reveal that the proposed algorithm outperforms min-min, max-min, suffrage, and First Come First Serve in terms of makespan. Kumar and Verma [109] presented an improved GA by using min-min and max-min techniques in the original GA for scheduling tasks in CC. In this improved GA, the initial population is generated by using min-min and max-min techniques. Makespan is considered as the fitness function. This improved GA is applied for two cases. In the first case, the number of VMs is kept constant and the number of cloudlets is varied, while in the second case, the number of cloudlets is fixed and the number of VMs is varied. Experimental results show that it performs better than the simple GA in terms of makespan and resource utilization. Shaminder *et al.* [110] proposed a modified GA by merging two existing scheduling algorithms into a standard GA for scheduling tasks in CC. In the proposed algorithm, the initial population is generated by using the output schedules of two algorithms (i.e., longest cloudlet to fastest processor and smallest cloudlet to fastest processor) and 8 random schedules. To achieve time minimization and compare it with existing heuristics, a fitness function is formulated for single-user jobs. Experimental results show that the proposed algorithm performs better than the standard GA in terms of response time.

Binh *et al.* [61] proposed a GA-based algorithm for solving task scheduling in cloud-FC. The proposed algorithm tries to achieve the optimal tradeoff between the execution time and operating costs by addressing different tasks in cloud-FC. Experimental results demonstrate that the proposed algorithm outperforms BA in terms of time-cost tradeoff. Wang *et al.* [111] studied an immune scheduling network-based method for task scheduling in FC. The proposed method uses forward and backward propagation in the ad hoc network along with the power of distributed schedulers to generate the optimized scheduler strategies to deal with computing node overloaded and achieve the optimal task finishing time reducing. Experimental results reveal that the proposed method performs better than the modified critical path, dynamic critical path, dominant sequence clustering, and GA.

2) EAs for Resource Allocation: Dolly [67] used DE for resource allocation in CC. The proposed method avoids premature convergence effectively, reduces the makespan, and increases the resource utilization. The proposed method outperforms PSO in terms of makespan, resource utilization, and load balancing. Lin and Zhong [112] presented a GA-based strategy for computing resource allocation in CC. They incorporated enhancements and local search into GA for solving computing resource allocation in CC. The computation time is focused in this work, which is an important factor in cloud manufacturing for fast response to users' requests. The proposed method is compared with other algorithms like PSO, BA, ACO, etc. Experimental results show the effectiveness of the proposed method in terms of makespan. Luo *et al.* [9] proposed a GA-based cashing placement strategy to minimize energy consumption in MEC. A joint optimization problem is formulated by considering energy consumption, backhaul capacities, and content popularity distributions. A GA is applied to solve this complicated joint optimization problem. Simulation results show that the proposed algorithm effectively

determines the near-optimal caching placement and obtains better performance in terms of energy efficiency compared with the conventional caching placement strategies.

3) EAs for Task Offloading: Chen and Yang [63] presented a data security strategy based on AIS in CC. They discussed the main factors affecting data security, introduced a Hadoop distributed file system, and applied AIS with negative selection and dynamic selection in CC. Simulation results show that the proposed strategy performs better than GA, PSO, and simulated annealing. Deng *et al.* [113] proposed a GA-based computation task offloading approach for MCC. This approach is designed for robust offloading decision to optimize execution time and energy consumption of mobile services. Numerical results demonstrate that with near-linear algorithmic complexity, the proposed approach can produce near-optimal solutions. Kaushik and Kumar [114] applied GA to a framework designed for elastic applications to reduce communication and computation energy by finding the optimum offloading solution. They focused on augmented execution of mobile applications and formulated an optimization problem by minimizing the cost function, which is the combination of communication energy and computation energy. Simulation results demonstrate that the proposed approach outperforms all mobile-side execution and all cloud-side execution.

Canali *et al.* [60] proposed a GA for service placement in FC. They studied mapping data streams over fog nodes and presented an optimization model. Then, a scalable heuristic based on GA is adopted to tackle this model. Experimental results verify the stability and performance of the proposed heuristic. Roman *et al.* [64] proposed an AIS-based strategy for the IoT system using edge technologies. In the proposed strategy, the requirements of immune system for IoT are analyzed and a security architecture is proposed to meet the demands of users. It makes a decision on the number and type of VMs deployed at the edge infrastructure. Farhoud *et al.* [65] proposed a new distributed and lightweight intrusion detection system in FC based on an AIS model. In this paper, the intrusion detection system is distributed to three layers: cloud, fog, and edge layers. Intrusion detection system, primary network traffic clustering, and detectors' training are performed in the cloud layer. Intrusion alerts are analyzed by using smart data concept in the fog layer. Detectors are deployed in edge clouds in the edge layer. Experimental results show the efficiency of the proposed system. Cheng *et al.* [10] proposed a fast heuristic algorithm based on GA for just-in-time code offloading for wearable computing. The proposed algorithm is compared with offloading nothing, offloading all to cloud, and simple greedy offloading. Numerical results show that the proposed algorithm outperforms the three competitors significantly. Tang *et al.* [11] suggested a task caching and migration strategy in MEC based on GA to satisfy the completion time constraints and the goal of minimizing energy consumption. They adopted a fine-grained task partitioning migration model to transform users' tasks into directed graphs with multiple subtasks. The task caching is proposed to further reduce the delay and energy consumption. Simulation results reveal that the proposed strategy can greatly reduce the energy consumption of end devices compared with all tasks being executed in the edge or end devices. Goudarzi *et al.* [115] proposed a GA-based algorithm for multi-site computation offloading in MCC. They modified genetic operators to reduce ineffective solutions and evaluated the efficiency of the proposed algorithm using graphs of real mobile applications. Experimental results demonstrate that the proposed algorithm outperforms other existing algorithms in terms of execution time, energy consumption, and weighted cost in a timely manner. Bozorgchenani *et al.* [116] proposed a multi-objective EA, i.e., non-dominated sorting GA, to find the optimal offloading decisions in MEC. They modeled the task offloading as a constrained multi-objective optimization problem that jointly minimizes the task processing delay and the energy consumption of mobile devices. Experimental results reveal that the proposed approach outperforms other existing approaches in terms of energy consumption and task processing delay.

4) EAs for Joint Issues: Wang *et al.* [68] studied a multi-unmanned aerial vehicle (UAV) enabled MEC, where the delay-sensitive task can be executed on the local device or one of UAVs. This paper designs a two-layer optimization method for jointly optimizing the deployment of UAVs and task scheduling, where DE and the greedy method are adopted at the upper layer and the lower layer, respectively. Guo *et al.* [117] presented a computation offloading model in the multi-access and multi-channel interference MEC. The offloading decision, channel allocation, and computation resource allocation are formulated as a mixed-integer nonlinear programming problem. A suboptimal algorithm, i.e., a GA-based computation algorithm, is proposed to solve this large-scale and NP-hard problem. Simulation results demonstrate that the proposed algorithm outperforms a PSO-based computation algorithm, a random computation algorithm, and a local computation algorithm in terms of energy consumption. Li and Zhu [118] proposed a joint optimization method based on GA to optimize offloading proportion, channel bandwidth, and mobile edge servers' computing resource allocation in MEC. Simulation results demonstrate that the proposed algorithm can effectively reduce the task completion time and guarantee fairness among users.

B. SIAs in CC and EC

This subsection discusses the works on the use of SIAs for solving issues in CC and EC, as summarized in Table III.

1) SIAs for Job Scheduling: Elina *et al.* [72] described a two-level cloud scheduler by using PSO, which operates under the IaaS model. They explored whether the use of a priority-based policy at the VM-level is suitable in an online cloud. Experimental results show that the proposed algorithm offers a good balance between throughput and response time. Guo *et al.* [73] proposed a PSO algorithm by incorporating a small position value rule to minimize processing cost in CC. Experimental results show that the proposed algorithm outperforms two other PSO-based algorithms in terms of convergence, processing time, and processing cost. Medhat *et al.* [78] presented an ACO algorithm for task scheduling in CC. At first, parameters with better values are determined for ACO through experiments. Then, ACO is evaluated on a set of instances with up to 1000 tasks. Wang

TABLE III
SIAs FOR ADDRESSING ISSUES IN CC AND EC

Issue	SIA	Metric	Paradigm	Reference
Job Scheduling	PSO	throughput, response time	CC	[72]
	PSO	makespan, cost	CC	[73]
	ACO	makespan	CC	[78]
	ACO	energy consumption, resource utilization	MCC	[79]
	ACO	resource utilization	CC	[152]
	ACO	energy consumption	MCC	[153]
	BA	makespan	CC	[83]
	BA	response time, execution time	CC	[84]
	CSA	setup cost, running cost	CC	[87]
	FA	energy consumption, resource utilization	CC	[91]
	FA	makespan	CC	[92]
	BFA	makespan, computation cost, resource utilization	CC	[95]
	BFA	cost, makespan	CC	[96]
	BA	execution time	FC	[85]
	CSA	response time, cost	CC & FC	[88]
	ACO	energy consumption, waiting time	CC	[80]
	ACO	energy consumption, resource utilization	CC	[154]
Resource Allocation	FA	resource utilization, cost	CC & FC	[93]
Task Offloading	PSO	energy consumption, latency	CC	[74]
	ACO	response time	MCC	[155]
Joint Issues	ACO	energy consumption	MEC	[81]

et al. [79] investigated MCC-assisted execution of a multi-task scheduling problem in a hybrid MCC architecture, and formulated it as an optimization problem with time constraints. Cooperative multi-task scheduling based on ACO is used to solve this optimization problem by considering task profit, task deadline, task dependence, node heterogeneity, and load balancing. Zhang and Zhang [152] applied an ACO algorithm for load balancing in a complex network for CC. The purpose of this study is to cope with the dynamic load balancing problem in open CC federation. Experimental results show that the proposed method performs better than SearchMax-SearchMin and classic ACO in terms of minimizing standard deviation and gains a more suitable distribution of the workloads on the whole cloud federation. Wei *et al.* [153] studied application scheduling in MCC with load balancing by using ACO. They presented a hybrid local mobile cloud model by extending the cloudlet architecture. The proposed model can select applications with the maximum profit and minimum energy consumption in a heavy load environment.

Hesabian *et al.* [83] presented a scheduling method in CC based on BA to dedicate the sources optimally. The proposed method behaves like a load balancing BA for small-scale systems in terms of makespan, while outperforms it in large-scale systems. Walaa *et al.* [84] proposed a BA for load balancing in CC, which distributes workloads of multiple network links to avoid under utilization and over utilization of resources. The proposed algorithm can minimize the response time and data center processing time by distributing workloads on different VMs based on the availability and load of each VM. It performs better than modified throttled and round robin algorithms in terms of average response time and execution time. Supacheep *et al.* [87] applied CSA for job scheduling in CC. They considered only static cases of job scheduling. The results of CSA are better than those of GA in most cases and it can find larger number of feasible solutions than GA. Kansal and Chana [91] proposed an energy-aware VM migration technique based on FA for CC, which migrates the maximally loaded VM to the least loaded active node while maintaining the performance and energy efficiency of the data centers. This technique is compared with ACO-based and FFD-based techniques in the Cloudsim simulator. Kaur and Sherma [92] applied FA to address workflow scheduling in CC, which reduces the execution time by efficient scheduling of workflow. The proposed FA performs better than other SIAs for workflow scheduling in terms of makespan. Verma *et al.* [95] proposed an improved BFA for scheduling tasks in CC. The performance of this algorithm is evaluated by using CloudSim toolkit. Experimental results demonstrate that it outperforms BFA and two other algorithms in terms of makespan, computation cost, and resource utilization. Jacob *et al.* [96] presented a BFA for resource scheduling in CC. In this work, a hyper-heuristic-based scheduling algorithm is used in the cloud system to map the resources in an efficient way. The proposed BFA reduces the cost and makespan and outperforms some existing algorithms like GA, ACO, a priority-based algorithm, and a Berger model-based algorithm.

Bitam *et al.* [85] proposed a BA for job scheduling in FC, in which two factors are considered: the CPU execution time and the allocated memory required by the overall tasks. The reliability and efficiency of the proposed BA are evaluated by performing a set of tests. It is superior to GA and PSO in terms of allocated memory and execution time. Nazir *et al.* [88] proposed a CSA for balancing load of users' requests in residential areas in cloud-FC. The performance of CSA is compared with that of some existing techniques like round robin and throttled algorithms. Pang *et al.* [80] designed an ACO algorithm to address dynamic energy management in the cloud data center. They used Petri net for analyzing scheduling process, and then a task-oriented resource allocation method is proposed to optimize the running time and energy consumption of the system. Liu *et al.* [154] proposed an algorithm called OEMAC based on ACO to minimize energy for VM placement in CC. OEMAC minimizes the number of active servers. Experimental results show that OEMAC outperforms conventional heuristics and

TABLE IV
FS FOR ADDRESSING ISSUES IN CC AND EC

Issue	FS	Metric	Paradigm	Reference
Job Scheduling	FL	response time, throughput, resource utilization	CC	[99]
	FL	latency	CC	[101]
	FL	latency, cost	CC	[156]
	FL	latency	CC	[157]
	FL	resource utilization	CC	[158]
Resource Allocation	FL	resource utilization, cost	CC	[100]
	FL	cost	CC	[102]
	FI	resource utilization	CC	[105]
Task Offloading	FL	security and privacy	CC	[103]
	FI	security and privacy	CC	[107]
	FI	security and privacy, execution time	MEC	[106]
Joint Issues	FL	resource utilization, response time	EC & CC	[159]
	FL	response time, processing time	CC	[160]

some other EAs in terms of saving energy, improving resource utilization, minimizing the number of active servers, and balancing different resources.

2) SIAs for Resource Allocation: Kanza *et al.* [93] utilized a FA for efficient resource allocation in cloud-FC, in which load balancing and cost reduction are considered.

3) SIAs for Task Offloading: Deng *et al.* [74] adopted discrete PSO to search the optimal offloading policy in cloud-enhanced small cell networks, with the aim of minimizing energy consumption under strict delay constraints. The energy-efficient task offloading problem is formulated as a constrained 0-1 programming problem. Experimental results show that the proposed approach outperforms local execution and conventional rough-granularity offloading policy in terms of energy saving. Bao *et al.* [155] proposed an ACO-based method for addressing computation offloading problem in MCC. Different rules are adopted to provide services to a large number of service requests.

4) SIAs for Joint Issues: Huang *et al.* [81] studied an ACO-based bilevel optimization approach for joint offloading decision and resource allocation in cooperative MEC to attain energy-efficient task execution under delay constraints. The effectiveness of the proposed approach is demonstrated by comparing it with four other algorithms.

C. FS in CC and EC

The works devoted to the use of FS in CC and EC are discussed in this subsection and summarized in Table IV.

1) FS for Job Scheduling: Anindita [99] proposed a new dynamic task scheduling approach in CC based on FL, which takes two inputs: the time required to complete the tasks in each VM and the number of requests received from each VM. These inputs produce an output i.e., id of VM which is assigned to the host. Experimental results show that the proposed algorithm outperforms First Come First Serve, round robin, and BA in terms of throughput, response time, and resource utilization. Srinivas *et al.* [101] proposed an efficient load balancing algorithm by using FL based on round robin load balancing technique for attaining better resource utilization in CC. For evaluating the balanced load through FL, two parameters are used: processor speed and assigned load of a VM. Numerical results reveal that the proposed algorithm outperforms round robin load balancing technique in terms of minimizing the processing and response time. Ragmani *et al.* [156] proposed an improved scheduling strategy based on FL in CC to evaluate processing time, response time, and total cost. FL uses the number of VMs, bandwidth, the number of processors, processor speed, the size of data, and request per user per hour as inputs. Then, the fuzzy controller calculates the global performance indicator. Issawi *et al.* [157] proposed an efficient adaptive load balancing algorithm in CC based on FL and round robin/random assignment, which consists of three main components: burst detector, load balancing algorithm, and fuzzifier. When a request is received in the data center, it is detected by the burst detector as normal or burst workload state. Then, a load balancing algorithm (round robin in burst state or random assignment in non-burst state) is selected, which assigns the received task to an appropriate VM based on the information provided by the fuzzifier. Experimental results show that the proposed algorithm reduces the response and processing time. Mondal *et al.* [158] used FL to improve QoS in CC by balancing the imposed load in the system. This paper considers the speed of processor and load as inputs and load balancing as output.

2) FS for Resource Allocation: Haratian *et al.* [100] proposed an adaptive and fuzzy resource management approach called AFRM for allocating resources in CC. In AFRM, the last resource values of each VM are collected through the environment sensors and sent to a fuzzy controller. Then, AFRM analyzes the received information to make a decision on how to reallocate the resources in each iteration of a self-adaptive control cycle. Experimental results show that AFRM outperforms rule-based and static-fuzzy approaches in terms of resource allocation efficiency, utility, the service level agreement violations, and cost. Wang *et al.* [102] proposed a strategy for resource allocation based on improved fuzzy clustering in CC. They first divided the set of resources in CC into different resource pools on the service level of users by using improved fuzzy clustering, and then generated a resource scheduling scheme by using a scheduling algorithm. The proposed strategy shows certain advantages in terms of the number of iterations and the accuracy of classification compared with other algorithms. Tao *et al.* [105] designed a containerized test environment in CC and developed a node selection algorithm based on FI for container deployment, where FI is applied to dynamically predict the

TABLE V
LBS FOR ADDRESSING ISSUES IN CC AND EC

Issue	LBS	Metric	Paradigm	Reference
Job Scheduling	RL	makespan	CC	[128]
	DRL	latency	CC	[136]
	DRL	energy consumption	MEC	[137]
Resource Allocation	RL	response time, resource utilization	CC	[129]
	RL	latency	FC	[126]
	RL	energy consumption, latency	EC	[130]
	DRL	latency	MEC	[138]
	DL	response time, resource utilization	EC	[161]
	DRL	response time	MEC	[162]
	DRL	resource utilization, makespan	MEC	[163]
Task Offloading	DRL	resource utilization, latency	MCC	[164]
	RL	resource utilization, latency	MCC	[165]
	DL	security and privacy	MEC	[8]
	DL	energy consumption, makespan	Cloudlet	[123]
	RL	energy consumption	MEC	[131]
	DRL	energy consumption, latency	MEC	[139]
	DRL	latency, cost	MEC	[140]
	RL	processing time, latency	MEC	[141]
	DRL	energy consumption	FC	[142]
	RL	energy consumption	MEC	[166]
	DRL	latency	MEC	[167]
	DRL	latency, energy consumption	MEC	[168]
	DRL	latency, energy consumption, computation cost	MEC	[169]
Joint Issues	DL	energy consumption, cost	MEC	[122]
	DRL	cost, energy consumption, latency	MEC	[143]
	DRL	latency, resource utilization	EC	[144]
	DRL	energy consumption, execution time	EC	[145]
	RL	security and privacy, energy consumption	MEC	[170]
	DRL	latency, energy consumption	MEC	[171]
	DRL	latency	MEC	[172]

most proper node for the deployment of the selected containers. Experimental results show that the proposed algorithm performs better than existing container deployment algorithms.

3) FS for Task Offloading: Ritu and Jain [103] presented a trust model in CC by using FL. The proposed model makes use of turnaround time, availability, and reliability for evaluating trust in CC. Qu and Buyya [107] proposed a cloud trust evaluation system by using hierarchical FI for service selection in CC. To facilitate service selection, this system evaluates the trust of clouds according to users' fuzzy QoS requirements and services' dynamic performance. Simulations and case studies demonstrate the effectiveness and efficiency of this system. Li *et al.* [106] designed an architecture in MEC to decouple the security functions with physical resources and developed a FI-based algorithm to find the optimal order of the required security functions. Numerical results show that the proposed algorithm performs better than a widely used fuzzy-based simple additive weighting algorithm in terms of inverted generational distance values and execution time.

4) FS for Joint Issues: Sonmez *et al.* [159] proposed an FL-based approach for workload orchestration in EC, where execution locations for incoming tasks from mobile devices are decided within an EC infrastructure. Simulation results demonstrate that the proposed approach performs better than other algorithms for the cases studied in terms of resource utilization and response time. Zulkar *et al.* [160] presented a FL-based dynamic load balancing algorithm in virtualized data centers, which can efficiently predict the VM where the next job is scheduled. They modeled the requirements of memory, bandwidth, and disk space using FL. Simulation results demonstrate that the proposed algorithm outperforms other scheduling algorithms in terms of response and processing time in the data centers.

D. LBS in CC and EC

In this subsection, we review the works on the use of LBS in CC and EC as summarized in Table V.

1) LBS for Job Scheduling: Peng *et al.* [128] proposed a RL-based mixed job scheduler scheme for CC, which considers accurate scaled CC environment and efficient job scheduling under VM resource and service level agreement constraints. The proposed scheme outperforms fast-fit, best-fit, min-min, and max-min scheduling schemes in terms of makespan. Lin *et al.* [136] presented a multi-resource cloud job scheduling strategy in CC based on current popular DRL and deep Q-network, which aims to reduce the average job completion time and average job slowdown. In this strategy, the convolutional NN is adopted to perceive the system resources, job state features, and RL decision-making capabilities to solve online awareness decision problems in CC. Based on the experimental results, the proposed strategy performs better than classical heuristic algorithms and converges faster than a standard policy gradient algorithm. Zhang *et al.* [137] combined a stacked auto-encoder with a Q-learning model to design a deep Q-learning model for energy-efficient scheduling in real-time systems. The main function of the stacked auto-encoder is to replace the Q-function for learning the Q-value. The proposed model can save 4.2%

energy compared with hybrid dynamic voltage and frequency scaling scheduling based on Q-learning for different sets of tasks.

2) LBS for Resource Allocation: Dutreilh *et al.* [129] studied RL for autonomic resource allocation in CC, in which proper initialization is adopted at the early stages and convergence speedups are applied in the learning phases. Nassar and Yilmaz [126] proposed a RL-based resource allocation algorithm in fog radio access networks. They formulated the resource allocation problem as an MDP in two alternative formulations: infinite-horizon MDP and finite-horizon MDP. Experimental results show that the proposed algorithm outperforms the fixed-threshold method. Liu *et al.* [130] proposed a RL approach based on ϵ-greedy Q-learning for resource allocation in EC. Experimental results show the effectiveness of the proposed approach in terms of minimizing energy consumption and latency. Yang *et al.* [138] proposed DRL-based resource allocation in MEC. They allocated computation resources by investigating different strategies in MEC networks that operate with finite block length codes to support low-latency communications. Simulation results show that the proposed algorithm outperforms the random and equal scheduling benchmarks. Luong *et al.* [161] proposed an optimal auction based on DL in the edge resource allocation. They constructed a multi-layer NN architecture based on analytical solution of the optimal auction. The NN first performs monotone transformations of the miners' bids. Then, allocation and conditional payment rules are calculated for the miners. Simulation results show that the proposed scheme can quickly converge to a solution. Wang *et al.* [162] proposed a DRL-based approach for smart resource allocations including computing resource allocation and network resource allocation in MEC. They considered two aspects: average service time minimization and resource allocation balancing. Experimental results reveal that the proposed approach outperforms the traditional open shortest path first algorithm. Xiong *et al.* [163] proposed a DRL-based approach for resource allocation of IoT in EC. They formulated the resource allocation problem as an MDP. They also proposed an improved deep Q-network algorithm for policy learning, where multiple replay memories are applied to separately store the experiences with small mutual influence. Simulation results show that the proposed algorithm outperforms the original deep Q-network algorithm in terms of convergence, and that the corresponding policy performs better than other policies regarding completion time.

3) LBS for Task Offloading: Quan *et al.* [164] proposed a two-layered RL algorithm for task offloading with a tradeoff between physical machine utilization rate and delay in MCC. The k-nearest neighbors algorithm divides the physical machines into many clusters. The first layer selects a cluster via learning the optimal policy, while the second layer learns an optimal policy to choose the optimal physical machine to execute the current offloaded task. Numerical results show that the proposed algorithm is faster than DRL when learning the optimal policy for task offloading. Sundar and Liang [165] proposed a game and learning approach for multi-user computation offloading in MCC. This paper discusses both online and offline computation task offloading from multiple users to a cloud or nearby cloud at the edge. The offline algorithm provides a better average solution while the online algorithm is much faster.

Chen *et al.* [8] proposed a DL-based model to detect security threats in MEC. This model uses unsupervised learning and location information to improve the detection process, which can detect malicious applications at the edge of a cellular network. Numerical results demonstrate that the proposed model outperforms softmax regression, decision tree, support vector machine, and random forest. Rani and Pounambal [123] proposed DL-based dynamic task offloading in mobile cloudlet, which considers energy consumption and execution time as objective metrics. The task computed on cloudlet or cloud server is divided into subtasks. Experimental results show that the proposed algorithm outperforms cloudlet-based dynamic task offloading in terms of energy consumption and completion time. Dinh *et al.* [131] proposed a RL-based computation offloading scheme in MEC to reduce energy consumption. They studied multi-user multi-edge-node computation offloading problem, and formulated it as a non-cooperative game where each user maximizes its own utility. Experimental results show that the proposed scheme outperforms local processing and random assignment. Meng *et al.* [139] proposed a DRL-based task offloading algorithm to minimize the mean slowdown of tasks and energy consumption of MEC. A new reward function is designed to optimize the tradeoff between average slowdown and average energy consumption. Numerical results reveal that the proposed algorithm performs better than the baseline algorithms such as all in MEC server, all in mobile device, and random in terms of average energy consumption and average slowdown.

Chen *et al.* [140] presented two DRL algorithms for optimizing computation offloading performance in virtual EC, where the stochastic computation offloading problem is formulated as an MDP. Numerical results show that the proposed algorithms outperform mobile execution, server execution, and greedy execution in terms of computation offloading performance. Zhang *et al.* [141] proposed a DRL-based task offloading scheme for vehicular edge computing networks (VECNs), in which the central cloud server is considered as a backup server due to its powerful computation capacities. The task offloading problem is formulated as a processing time minimization problem with delay constraints. Ning *et al.* [142] studied DRL for intelligent Internet of vehicles. They constructed an offloading framework consisting of three layers (i.e., cloudlet, RodeSide Units, and fog nodes) to minimize the total energy consumption under the delay constraint. The proposed method outperforms the baseline algorithms in terms of average energy consumption. Ranadheera *et al.* [166] proposed a computation offloading approach based on game theory and RL in MEC to reduce energy consumption.

Wang *et al.* [167] proposed a DRL-based method for computation offloading in MEC. The offloading problem in MEC is formulated as an MDP and the S2S neural network is designed to represent the offloading policy. Simulation results show that the proposed method performs better than two heuristic baselines in terms of latency, and can obtain nearly optimal results while having polynomial time complexity. Chen *et al.* [168] proposed

a DRL-based approach for performance optimization in MEC. They considered MEC for a representative mobile user in an ultra-dense network, where one of multiple base stations can be selected for computation offloading. They modeled an optimal computation offloading policy as an MDP and developed a deep Q-network-based strategic computation offloading algorithm to learn the optimal policy without having any priori knowledge of the dynamic statistics. Chen and Wang [169] proposed a DRL-based approach for decentralized computation offloading in MEC to minimize the long-term average computation cost in terms of power consumption and buffering delay. They adopted continuous action space-based DRL to learn efficient computation offloading policies independently at each mobile user. Experimental results demonstrate that the proposed approach outperforms conventional deep Q-network-based discrete power control strategy and some other greedy strategies in terms of computation cost.

4) LBS for Joint Issues: Huang *et al.* [122] proposed distributed DL-based offloading in MEC. They formulated joint offloading decision and bandwidth allocation as a mixed-integer programming problem. Multiple parallel deep NNs are used to generate offloading decisions. Numerical results reveal that the proposed algorithm outperforms deep Q-network in terms of total cost and energy consumption. Huang *et al.* [143] further studied the same joint issue in multi-user MEC by adopting the idea of deep Q-network. Experimental results show that the proposed algorithm performs better than the MUMTO algorithm in [177] in terms of overall cost, energy consumption, and delay. Liu *et al.* [144] formulated offloading and resource allocation in VECNs as a semi-Markov process, by considering stochastic vehicle traffic, dynamic computation requests, and time-varying communication conditions. Two RL methods, i.e., a Q-learning-based method and a DRL method, are designed to get the optimal policies for computation offloading and resource allocation. Ning *et al.* [145] constructed an intelligent offloading system for VECNs by using DRL. They used Markov chains to model communication and computation states. To improve users' quality of experience, task scheduling and resource allocation are formulated as a joint optimization problem. To schedule offloading requests and allocate network resources, this paper designs a two-sided matching scheme and a DRL approach. Experimental results show that the proposed method outperforms Q-learning, a greedy method, local computing, and deep Q-network in terms of quality of experience and execution time. Xiao *et al.* [170] applied RL to provide secure offloading to the edge nodes against jamming attacks in MEC. Lightweight authentication and secure collaborative cashing schemes are presented for securing data privacy. Simulation results show that the proposed RL-based secure solution can effectively enhance the security and user privacy of MEC and protect MEC against various smart attacks with low overhead. Li *et al.* [171] proposed a DRL-based computation offloading and resource allocation scheme for MEC. In order to minimize the sum cost of delay and energy consumption for all users' equipments in MEC, they jointly optimized the offloading decision and computing resource allocation. Simulation results reveal that the proposed scheme performs better than other baselines. Huang *et al.* [172] proposed a DRL-based method for online computation offloading in wireless powered MEC networks to maximize the weighted sum computation rate with binary computation offloading. They optimally adapted task offloading decisions and wireless resource allocations to the time-varying wireless channel conditions. Simulation results demonstrate that the proposed algorithm achieves similar near-optimal performance as existing benchmark methods but reduces the CPU execution latency by more than an order of magnitude.

E. HS in CC and EC

This section discusses the works on the use of HS for solving issues in CC and EC, which are summarized in Table VI.

1) HS for Job Scheduling: Manasrah and Ali [148] proposed a hybrid GA-PSO algorithm for workflow task scheduling in CC. It outperforms GA, PSO, and some other CI techniques in terms of total execution time of the workflow tasks. Liu *et al.* [149] proposed a hybrid GA-ACO task scheduling algorithm in CC, which takes advantage of fast convergence from ACO and global search ability of GA. Simulation results show that it outperforms GA and ACO in terms of task execution time. Rashidi and Sharifian [150] presented a hybrid GA-ACO algorithm for task scheduling in MCC. Experimental results show that it outperforms queue-based round robin, queue-based random, and queue-based weighted round robin assignment algorithms in terms of average completion time, total energy consumption of mobile devices, and the number of dropped tasks. Moganarangan *et al.* [151] combined ACO with CSA to address job scheduling in CC. The proposed algorithm performs better than ACO in terms of energy consumption and makespan.

2) HS for Resource Allocation: Vimal *et al.* [173] proposed a hybridization of RL and multi-objective ACO to enhance resource allocation in MEC for industrial IoT. The proposed algorithm allocates resources accurately and optimally for users in MEC. Experimental results show that the proposed algorithm outperforms GA and BA in terms of throughput.

3) HS for Task Offloading: Saljoughi *et al.* [174] presented a hybrid NN-PSO algorithm to detect intrusions and attacks in CC. For extraction of the optimal weights of NN, the weights of NN are optimized by using PSO. The proposed method obtains better outcomes than the simple NN on NSL-KDD and KDD-CUP databases.

4) HS for Joint Issues: Guo *et al.* [175] studied the energy-efficient computation offloading management scheme based on hybrid GA-PSO in MEC with small cell networks. They jointly optimized computation offloading, spectrum, power, and computation resource to minimize the energy consumption of all users' equipments. They presented the computation offloading model and formulated the problem as a mixed-integer nonlinear programming problem. Simulation results show that the proposed algorithm performs better than other baseline algorithms. Jiang *et al.* [176] presented a hybrid DL-PSO-FL algorithm for online offloading to minimize the energy consumption of users' equipments in a hybrid MEC network. In the proposed algorithm, FL is used to locate the ground vehicles and UAVs, PSO is used to solve the mixed-integer nonlinear programming problem and provides high-quality samples to DNN, and DL is applied to make the task admission and resource allocation decision in real time. Simulation results reveal that the proposed

TABLE VI
HS FOR ADDRESSING ISSUES IN CC AND EC

Issue	HS	Metric	Paradigm	Reference
Job Scheduling	GA-PSO	makespan, cost, execution time	CC	[148]
	GA-ACO	execution time	CC	[149]
	GA-ACO	energy consumption, makespan, resource utilization	MCC	[150]
	ACO-CSA	energy consumption, makespan	CC	[151]
Resource Allocation	RL-ACO	throughput	MEC	[173]
Task Offloading	NN-PSO	security and privacy	CC	[174]
Joint Issues	GA-PSO	energy consumption	MEC	[175]
	DRL-PSO-FL	energy consumption	MEC	[176]

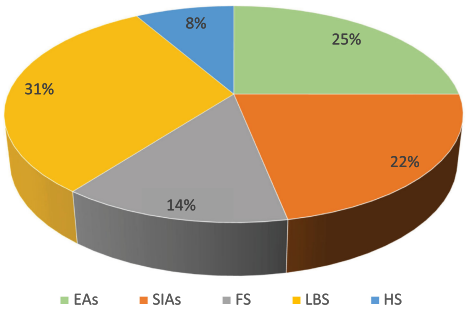


Fig. 5. Overall percentage distributions of CI techniques for addressing issues in CC and EC.

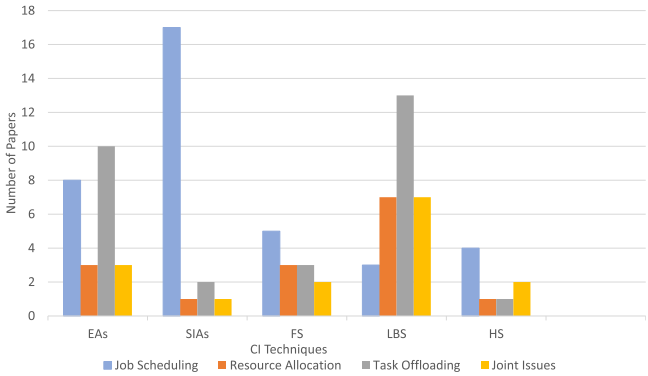


Fig. 6. Suitability of CI techniques for addressing issues in CC and EC.

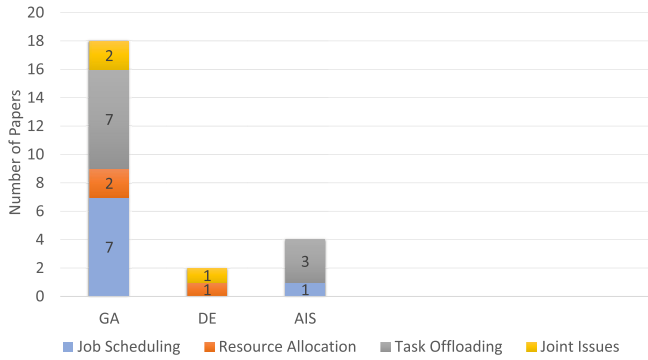


Fig. 7. Distribution of EAs in CC and EC.

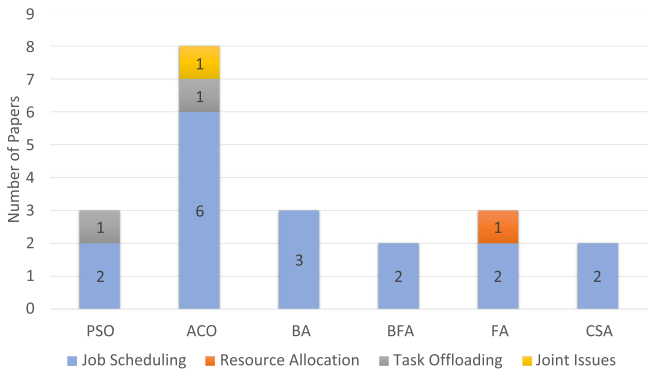


Fig. 8. Distribution of SIAs in CC and EC.

algorithm reduces the CPU time by more than several orders of magnitude.

V. DISCUSSION AND FUTURE RESEARCH TRENDS

Recent literature shows that researchers have paid much attention to the innovative use of CI techniques to address issues in CC and EC, such as job scheduling, resource allocation, task offloading, and joint issues. However, there are still open challenges for researchers to tackle. This section provides the statistics which reflect the status-quo of CI for CC and EC, and points out future research trends.

A. Discussion

Fig. 5 presents the overall percentage distributions of CI techniques mentioned in Fig. 4 for solving issues in CC and EC. Fig. 6 presents all issues, CI techniques, and the number of papers found for each issue in CC and EC. Furthermore, we elaborate on the main research findings derived from Section IV.

1) EAs for CC and EC: From Fig. 5, EAs are the second most commonly used techniques among all types of CI techniques. Most of the works found in this survey are on the use of EAs for job scheduling and task offloading as shown in Fig. 7. Furthermore, the most frequently used EA in CC and EC is GA as shown in Fig. 7. Fig. 7 also presents the distribution of EAs, which indicates the suitability of each kind of EAs for each issue in CC and EC.

2) SIAs for CC and EC: SIAs have been used in 22% papers compared with other CI techniques as shown in Fig. 5. We can observe from Fig. 8 that job scheduling is the hottest issue addressed by SIAs and ACO is the most frequently used SIA in CC and EC. Fig. 8 also shows the distribution of SIAs in the context of issues in CC and EC.

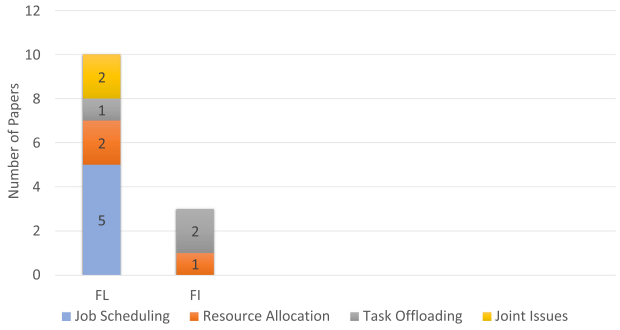


Fig. 9. Distribution of FS in CC and EC.

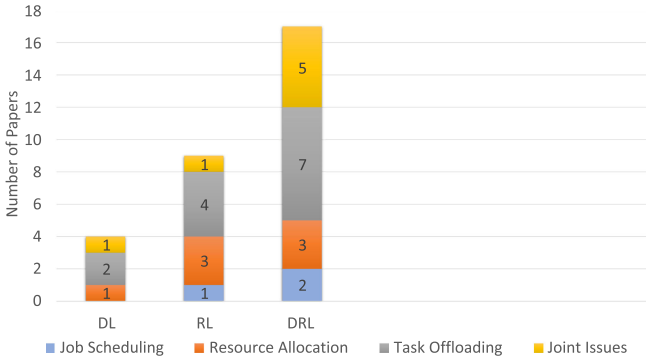


Fig. 10. Distribution of LBS in CC and EC.

3) FS for CC and EC: It can be seen from Fig. 5 that 14% papers employ FS to address issues in CC and EC. FS has been frequently used for job scheduling, resource allocation, and task offloading as shown in Fig. 9. Fig. 9 presents the distribution of FS in the context of issues in CC and EC.

4) LBS for CC and EC: LBS is the most commonly studied CI technique in CC and EC as shown in Fig. 5. In particular, it has been used more intensively for task offloading, resource allocation, and joint issues in CC and EC from Fig. 10. In addition, DRL is the most frequently used LBS as shown in Fig. 10. Fig. 10 gives the distribution of LBS in CC and EC.

5) HS for CC and EC: HS has not been utilized frequently (only 8% papers) in CC and EC as depicted in Fig. 5. Specifically, half of papers related to HS study on job scheduling as shown in Fig. 6.

Remark: Based on the above discussion, we would like to give more details to understand what kinds of CI techniques are good at solving what kinds of issues:

- GA is the best choice for solving job scheduling and task offloading among all EAs [10], [61], [108], [175]. The reasons are twofold: 1) it is the most popular and frequently used EA paradigm; and 2) it has various crossover and mutation operators that can deal with different optimization problems, such as discrete and continuous optimization problems.
- ACO is the best SIA for job scheduling among all the issues studied in this paper [79], [152], [153], due to the following reasons: 1) it can consider/incorporate multiple scheduling targets/metrics, such as load balancing, energy consumption, makespan, and cost; 2) each variable in the solution obtained by ACO is generated one by one; thus, jobs can be assigned to VMs one by one to optimize the scheduling process; and 3) it can ensure the fast convergence and good performance by balancing the exploration of new solutions and exploitation of accumulated experience about the problem [154].
- Compared with FI, FL is better for solving job scheduling, resource allocation, and joint issues [99], [101], since it is flexible, easy to understand, compatible with the uncertainty of CC and EC parameters as well as users' behaviors, and can deal with imprecise data and complex problems with several variables.
- DRL performs the best among all LBS for resource allocation, task offloading, and joint issues [145], [161], [163], [171]. The superiority of DRL can be summarized as follows: 1) it can automatically adapt and customize itself according to users' requirements [162]; 2) it can discover/learn new knowledge from large databases; 3) it can develop models that are difficult and expensive to be designed manually due to the requirements of specific skills; and 4) it has a strong ability to handle complex problems by efficiently learning from experiences.

B. Future Research Trends

1) From CC and EC Perspective:

- Many-Metric Formulation: More metrics need to be considered at the same time in CC and EC to create the trade-off among them. For example, in the future, we may consider the trade-off among latency, cost, energy consumption, security, and privacy to satisfy QoS of users.
- Edge Node Allocation: Due to the distributed nature of EC, edge clouds that offer services across diverse geolocation and regions are difficult to be allocated [178]. In the future, efficient service discovery protocols are needed to design such that users can identify and locate the relevant service providers to meet their demands.
- Real-Time Optimization: For many edge application scenarios, the service environments are of high dynamics [60] and it is hard to correctly foresee future events. Thus, it would require the remarkable capabilities of online edge resource orchestration and provisioning to continuously handle massive dynamic workloads and tasks. In the future, real-time resource allocation is required to fulfill dynamic task demands.
- Decentralized Trust: The open nature of EC leads to the decentralized trust, e.g., services provided by different edge entities must be secured and trustworthy. Thus, efficient security mechanisms are required to ensure users' authentication, data integrity, and mutual platform verification for EC [20]. Also, novel secure routing schemes and trust network topologies are critical for EC. In addition, end devices would generate a large volume of data at the network edge, which can be privacy-sensitive since they may

contain users' location data, health status, personal activities records or other sensitive information [49]. Therefore, feasible paradigms are needed to secure data sharing to minimize the privacy leakage.

- Hardware Constraints in EC: Due to hardware constraints, unlike CC, EC cannot support heavy-weight software [49]. Big data analysis and data warehousing will never be feasible with existing EC because of the increasing number of duplication of small software in market. Therefore, users are facing difficulties in looking for a trusted edge provider among edge providers with a lack of standardized framework. Thus, developing software and hardware for handling computation offloading from the edge cloud is a critical issue to be addressed.
- Migration of EC Applications: Migration of EC applications among different edge clouds is a challenging issue that can help to balance loads or accommodate users' movements and minimize the end-to-end latency of users [35]. In the future, efficient techniques are required based on system measurements and experiments to handle the migration challenges.
- Interoperability and Collaboration of CC and EC: Thanks to EC, users are able to process latency-sensitive applications at the network edge [179]. However, handling an increasing number of multiple services is still a big challenge. Some tasks may be redirected to the central clouds for further processing. Thus, it is important to develop effective architectures and efficient algorithms to facilitate the above collaboration.
- Heterogeneity: The future edge and cloud networks may present a huge level of heterogeneity, i.e., we may come across various kinds of users, ranging from smartphones, smartwatches, sensor nodes, and intelligent cameras [180]. This may cause the instability of the networks and, therefore, efficient solutions are highly required [181].
- Edge Intelligence: There is a growing trend to process data at edge clouds because of privacy and other concerns [20]. However, due to the limited resources in edge clouds, some computation-intensive tasks such as machine learning models may not be trained at edge clouds without any proper modification. Therefore, in the future, it is important to study how to execute or train machine learning models in edge clouds. Federated learning has been proposed to address this issue but more attempts are expected in the future.
- Blockchain-Assisted CC and EC: Blockchain has attracted much attention from both academia and industry [182], [183]. It is also interesting to design the blockchain-based networks for CC and EC. Three aspects can be considered here: 1) blockchain can be applied to enhance the privacy in the communication and computing resource sharing; 2) blockchain can be very useful in terms of improving the price and security of CC and EC; and 3) as blockchain may need huge computing resources to run, CC and EC can provide computing resources to blockchain-related tasks or applications.

2) From CI Technique Perspective:

- Execution Time: Most CI techniques like EAs need a long time for searching the optimal solution in CC and EC. In the future, we may find a way to decrease the execution time.
- Convergence: Sometimes, it is not easy to prove the convergence in CI techniques. However, in CC and EC, it is important to have an algorithm with convergence guarantee. Therefore, a future trend is to design some CI techniques with good convergence performance.
- Multi/Many-Objective Optimization: Two or more objectives need to be addressed in CC and EC to meet the service level agreements of users. Therefore, a future trend may be to design multi/many-objective CI techniques to address multi/many-objective optimization problems in CC and EC.
- Constraint-Handling Issue: Since most CI techniques cannot handle constraints effectively, we can incorporate some constraint-handling techniques into CI techniques to address constrained optimization problems in CC and EC.
- CI Techniques for Security and Privacy: Directly sharing data among multiple edge nodes may run a high risk of privacy leakage. Therefore, federated learning paradigms can be applied to secure data sharing by training distributed data such that the original data sets can be kept in their source nodes and the edge AI model parameters can be shared among different nodes.
- AI Edge Models: Instead of utilizing the existing resource-intensive AI models in CC and EC, we can design a resource-aware edge AI model. For example, methods like CI techniques can be adopted to efficiently search over the AI model design parameter space by taking into account the impact of hardware resource constraints on the performance metrics such as execution latency, security, and energy overhead.
- CI Techniques for Migration of EC Applications: Sharing and migrating applications among edge clouds is a challenging task, as we have to consider the load balancing and reduce the latency for all tasks. LBS and FS may be suitable candidates for migration of EC applications.
- CI Techniques for Interoperability and Collaboration of CC and EC: Multiple latency-sensitive requests may arrive at the same time. Therefore, fast CI techniques (such as LBS and FS) along with prediction algorithms are potential for the collaboration between the central and edge clouds.
- Hybrid Versions: Hybrid CI techniques have shown great potential in solving complex optimization problems; however, few papers have studied them for addressing issues in CC and EC. A future trend may be to design hybrid CI techniques. For instance, FS is suitable for job scheduling while LBS is competitive for task offloading. In the future, we may combine FS and LBS to address joint issues. Similarly, more hybrid CI techniques can be developed to deal with energy consumption, completion time, and security in CC and EC.

VI. Conclusion

In this survey paper, we reviewed the applications of CI techniques to four critical issues in CC and EC: job scheduling, resource allocation, task offloading, and joint issues. We commenced with rudimentary concepts of CC and EC along with critical issues and metrics, and then focused on five categories of CI techniques used in CC and EC: EAs, SIAs, FS, LBS, and HS. Subsequently, diverse designed approaches relying on CI techniques were reviewed in the context of CC and EC. In addition, the statistics about the status-quo of CI for CC and EC were provided based on the works collected in this survey paper. We found that LBS was intensively used in CC and EC, followed by EAs and SIAs. However, FS and HS were not fully utilized in CC and EC. Finally, we pointed out some challenges and future research trends in CI, CC, and EC.

References

[1] M. Armbrust *et al.*, "Above the clouds: A Berkeley view of cloud computing," EECS Department, University of California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-28, Feb. 2009.

[2] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, May. 2010.

[3] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, Tech. Rep. 800-145, Sep. 2011.

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

[5] N. Wingfield, "Amazon's profits grow more than 800 percent, lifted by cloud services," New York Times, New York, NY, USA, Jul. 2016. [Online]. Available: https://www.nytimes.com/2016/07/29/technology/amazon-earnings-profit.html?_r=0

[6] I. Bojanova and A. Samba, "Analysis of cloud computing delivery architecture models," in *Proc. IEEE Workshops Int. Conf. Adv. Inform. Netw. Appl.*, 2011, pp. 453–458.

[7] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[8] Y. Chen, Y. Zhang, S. Maharjan, M. Alam, and T. Wu, "Deep learning for secure mobile edge computing in cyber-physical transportation systems," *IEEE Netw.*, vol. 33, no. 4, pp. 36–41, Jul./Aug. 2019.

[9] Z. Luo, M. LiWang, Z. Lin, L. Huang, X. Du, and M. Guizani, "Energy-efficient caching for mobile edge computing in 5G networks," *Appl. Sci.-basel J.*, vol. 7, pp. 1–13, 2017.

[10] Z. Cheng, P. Li, J. Wang, and S. Guo, "Just-in-time code offloading for wearable computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 1, pp. 74–83, Mar. 2015.

[11] L. Tang, B. Tang, L. Kang, and L. Zhang, "A novel task caching and migration strategy in multi-access edge computing based on the genetic algorithm," *Future Internet*, vol. 11, no. 8, pp. 1–14, 2019.

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

[13] D. T. Hoang, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, pp. 1587–1611, 2013.

[14] R. M. Singh, S. Paul, and A. Kumar, "Task scheduling in cloud computing: Review," *Int. J. Comput. Sci. Inf Technol.*, vol. 5, no. 6, pp. 7940–7944, 2014.

[15] A. A. Helen and B. V. RebeccaJeya, "A survey on quality of service in cloud computing," *Int. J. Comput. Trends Technol.*, vol. 27, no. 1, pp. 58–63, Sep. 2015.

[16] P. Rajeswari and K. Jayashree, "Survey on QoS metrics and ranking in cloud services," *Int. J. Eng. Technol.*, vol. 7, no. 1.3, pp. 146–149, 2018.

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

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

[19] F. Fatemi Moghaddam, M. Ahmadi, S. Sarvari, M. Eslami, and A. Golkar, "Cloud computing challenges and opportunities: A survey," in *Proc. 1st Int. Conf. Telematics Future Gener. Netw.*, May 2015, pp. 34–38.

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

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

[22] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[23] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou, "A survey on edge computing systems and tools," *Proc. IEEE*, vol. 107, no. 8, pp. 1537–1562, Aug. 2019.

[24] K. Peng, V. Leung, X. Xu, L. Zheng, J. Wang, and Q. Huang, "A survey on mobile edge computing: Focusing on service adoption and provision," *Wireless Commun. and Mobile Comput.*, vol. 2018, pp. 8 267 838:1–8 267 838:16, 2018.

[25] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 2:1–2:23, Feb. 2019.

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

[27] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for Internet of things realization," *IEEE Commun. Surv. Tut.*, vol. 20, no. 4, pp. 2961–2991, Oct.–Dec. 2018.

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

[29] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1628–1656, Jul.–Sep. 2017.

[30] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proc. 10th Int. Conf. Intell. Syst. Control*, Jan. 2016, pp. 1–8.

[31] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 125:1–125:36, Oct. 2019.

[32] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 63:1–63:33, Jul. 2015.

[33] M. Guzek, P. Bouvry, and E. Talbi, "A survey of evolutionary computation for resource management of processing in cloud computing," *IEEE Comput. Intell. Mag.*, vol. 10, no. 2, pp. 53–67, May 2015.

[34] P. Chopra and R. Bedi, "Applications of fuzzy logic in cloud computing: A review," *Int. J. Scientific Res. Eng. Technol.*, vol. 6, pp. 1083–1086, 2017.

[35] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[36] J. Wu, Z. Zhang, Y. Hong, and Y. Wen, "Cloud radio access network (C-RAN): A primer," *IEEE Netw.*, vol. 29, no. 1, pp. 35–41, Jan. 2015.

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

[38] M. Satyanarayanan *et al.*, "Edge analytics in the Internet of Things," *IEEE Pervasive Comput.*, vol. 14, no. 2, pp. 24–31, Apr.–Jun. 2015.

[39] Y. Gao, W. Hu, K. Ha, B. Amos, P. Pillai, and M. Satyanarayanan, "Are cloudlets necessary?" School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. CMU-CS-15-139, 2015.

[40] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of things," in *Proc. First Edition MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[41] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[42] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *Wireless Algorithms, Systems, and Applications*, K. Xu and H. Zhu, Eds. Berlin, Germany: Springer International Publishing, 2015, pp. 685–695.

[43] W. Hu *et al.*, "Quantifying the impact of edge computing on mobile applications," in *Proc. 7th ACM SIGOPS Asia-Pacific Workshop Syst.*, 2016, pp. 5:1–5:8.

[44] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[45] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for iot systems: A computation offloading game," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3246–3257, Aug. 2018.

[46] Y. Lan, X. Wang, C. Wang, D. Wang, and Q. Li, "Collaborative computation offloading and resource allocation in cache-aided hierarchical edge-cloud systems," *Electronics*, vol. 8, no. 12, pp. 1–30, 2019.

[47] E. Ahvar, A. Orgerie, and A. Lbre, "Estimating energy consumption of cloud, fog and edge computing infrastructures," *IEEE Trans. Sustainable Comput.*, to be published, doi: 10.1109/TSUSC.2019.2905900.

[48] R. Basedia and M. Kumbhkar, "Cloud computing security issues and challenges," *Int. J. Innovative Res. Comput. Commun. Eng.*, vol. 4, no. 4, pp. 6733–6736, Apr. 2016.

[49] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing fog computing for Internet of things applications: Challenges and solutions," *IEEE Commun. Surv. Tut.*, vol. 20, no. 1, pp. 601–628, Jan.–Mar. 2018.

[50] A. Al-Shaikh, H. Khattab, A. Sharieh, and A. Sleit, "Resource utilization in cloud computing as an optimization problem," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 6, pp. 336–342, 2016.

[51] M. Jelassi, C. Ghazel, and L. A. Sadane, "A survey on quality of service in cloud computing," in *Proc. 3rd Int. Conf. Frontiers Signal Process.*, Sep. 2017, pp. 63–67.

[52] R. Raju, R. G. Babukarthik, D. Chandramohan, P. Dhavachelvan, and T. Vengattaraman, "Minimizing the makespan using hybrid algorithm for cloud computing," in *Proc. 3rd IEEE Int. Advance Comput. Conf.*, Feb. 2013, pp. 957–962.

[53] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE Symp. Comput. and Commun.*, Jul. 2012, pp. 59–66.

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

[55] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Berlin, Germany: Springer, 2003.

[56] J. Holland, "Erratum: Genetic algorithms and the optimal allocation of trials," *SIAM J. Comput.*, vol. 3, no. 4, pp. 326–326, 1974.

[57] B. Hu, X. Sun, Y. Li, and H. Sun, "An improved adaptive genetic algorithm in cloud computing," in *Proc. 13th Int. Conf. Parallel Distrib. Comput., Appl. and Technol.*, Dec. 2012, pp. 294–297.

[58] M. Agarwal and G. M. S. Srivastava, "A genetic algorithm inspired task scheduling in cloud computing," in *Proc. Int. Conf. Comput., Commun. Autom.*, Apr. 2016, pp. 364–367.

[59] J. Liu, X.-G. Luo, X.-M. Zhang, F. Zhang, and B.-N. Li, "Scheduling model for cloud computing based on multi-objective genetic algorithm," *IJCSI Int. J. Comput. Sci. Issues*, vol. 10, no. 3, pp. 134–139, 2013.

[60] C. Canali and R. Lancellotti, "GASP: Genetic algorithms for service placement in fog computing systems," *Algorithms*, vol. 12, no. 10, pp. 1–19, 2019.

[61] H. T. T. Binh, T. T. Anh, D. B. Son, P. A. Duc, and B. M. Nguyen, "An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment," in *Proc. 9th Int. Symp. Inform. Commun. Technol.*, 2018, pp. 397–404.

[62] J. Timmis, A. Hone, T. Stibor, and E. Clark, "Theoretical advances in artificial immune systems," *Theoretical Comput. Sci.*, vol. 403, no. 1, pp. 11–32, 2008.

[63] J. Chen and D. Yang, "Data security strategy based on artificial immune algorithm for cloud computing," *Appl. Math. Inform. Sci.*, vol. 7, no. 1L, pp. 149–153, 2013.

[64] R. Roman, R. Rios, J. A. Onieva, and J. Lopez, "Immune system for the Internet of things using edge technologies," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4774–4781, Jun. 2019.

[65] H. Farhoud, A. Payam, Vahdani, P. Juha, H. Timo, and T. Hannu, "An intrusion detection system for fog computing and IoT based logistic systems using a smart data approach," *Int. J. Digit. Content Technol. Appl.*, vol. 10, no. 5, pp. 34–46, Dec. 2016.

[66] R. Storn and K. Price, "Differential evolution a simple evolution strategy for fast optimization," *Dr Dobbs*, vol. 22, no. 4, pp. 18–24, 1997.

[67] E. S. Dolly, "Efficiently resource allocation in cloud scheduling using differential evolution," *Int. J. Innovative Res. Comput. Commun. Eng.*, vol. 5, pp. 10 232–10 240, 2017.

[68] Y. Wang, Z. Ru, K. Wang, and P. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2019.2935466.

[69] J. Kacprzyk and W. Pedrycz, *Springer Handbook of Computational Intelligence*. Berlin, Germany: Springer, 2015.

[70] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, Nov. 1995, vol. 4, pp. 1942–1948.

[71] K. Parsopoulos and M. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Comput.*, vol. 1, no. 2, pp. 235–306, Jun. 2002.

[72] P. Elina, M. Cristian, and G. G. Carlos, "Dynamic scheduling based on particle swarm optimization for cloud-based scientific experiments," *Latin Amer. Center Comput. Stud.; CLEI Electron. J., CLEI Electron. J.*, vol. 14, no. 1, pp. 1–14, Apr. 2014.

[73] L. Guo, S. Zhao, S. Shen, and C. Jiang, "Task scheduling optimization in cloud computing based on heuristic algorithm," *J. Netw.*, vol. 7, pp. 547–553, 2012.

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

[75] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Politecnico di Milano, Milano, Italy, Tech. Rep. 91-016, 1991.

[76] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern., Part B (Cybern.)*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[77] C. Blum, "Ant colony optimization: Introduction and recent trends," *Phys. Life Reviews*, vol. 2, no. 4, pp. 353–373, 2005.

[78] T. Medhat, E.-S. Ashraf, K. Arabi, and T. Fawzy, "Cloud task scheduling based on ant colony optimization," *Int. Arab J. Inform. Technol.*, vol. 12, no. 2, pp. 129–137, Mar. 2015.

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

[80] S. Pang, W. Zhang, T. Ma, and Q. Gao, "Ant colony optimization algorithm to dynamic energy management in cloud data center," *Math. Problems Eng.*, vol. 2017, pp. 1–10, 2017.

[81] P. Huang, Y. Wang, K. Wang, and Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2019.2916728.

[82] D. Pham, A. Ghanbarhh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, "The Bees Algorithm, technical note," Manufacturing Engineering Center, Cardiff University, Cardiff, UK, Tech. Rep. MEC 0501, 2005.

[83] N. Hesabian, H. Haj, and S. Javadi, "Optimal scheduling in cloud computing environment using the Bee algorithm," *Int. J. Comput. Netw. Commun. Security*, vol. 3, no. 6, pp. 253–258, Jun. 2015.

[84] H. Walaa, N. Heba, and R. Rawya, "Honey bee based load balancing in cloud computing," *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 12, pp. 5694–5711, 2017.

[85] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Inf. Syst.*, vol. 12, pp. 1–25, 2017.

[86] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *Proc. World Congr. Nature Biol. Inspired Comput.*, Dec. 2009, pp. 210–214.

[87] A. Supacheep and M. Toshiyuki, "Cuckoo search algorithm for job scheduling in cloud systems," *IEICE Trans. Fundamentals Electronics, Commun. Comput. Sciences*, vol. E98.A, no. 2, pp. 645–649, 2015.

[88] S. Nazir, S. Shafiq, Z. Iqbal, M. Zeeshan, S. Tariq, and N. Javaid, "Cuckoo optimization algorithm based job scheduling using cloud and fog computing in smart grid," in *Advances in Intelligent Networking and Collaborative Systems*, F. Xhafa, L. Barolli, and M. Greguš, Eds. Berlin, Germany: Springer International Publishing, 2019, pp. 34–46.

[89] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*. London, UK: Luniver Press, 2008.

[90] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 2, pp. 78–84, 2010.

[91] N. J. Kansal and I. Chana, "Energy-aware, virtual machine migration for cloud computing - A firefly optimization approach," *J. Grid Comput.*, vol. 14, pp. 327–345, 2016.

[92] A. Kaur and S. Sharma, "Workflow scheduling in cloud computing environment using Firefly algorithm," *Int. J. Comput. Sci. and Technol.*, vol. 8, pp. 73–76, 2017.
[93] K. Hassan, N. Javaid, F. Zafar, S. Rehman, M. Zahid, and S. Rasheed, "A cloud fog based framework for efficient resource allocation using firefly algorithm," in *Advances on Broadband and Wireless Computing, Communication and Applications*, L. Barolli, F.-Y. Leu, T. Enokido, and H.-C. Chen, Eds. Berlin, Germany: Springer International Publishing, 2019, pp. 431–443.
[94] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Syst. Mag.*, vol. 22, no. 3, pp. 52–67, Jun. 2002.
[95] J. Verma, S. Sobhanayak, S. Sharma, A. Kumar-Turuk, and B. Sahoo, "Bacteria foraging based task scheduling algorithm in cloud computing environment," in *Proc. Int. Conf. Comput., Commun. Autom.*, 2017, pp. 777–782.
[96] L. Jacob, V. Jeyakrishanan, and P. Sengottuvelan, "Resource scheduling in cloud using bacterial foraging optimization algorithm," *Int. J. Comput. Appl.*, vol. 92, no. 1, pp. 14–20, Apr. 2014.
[97] L. A. Zadeh, "Fuzzy logic," *Computer*, vol. 21, no. 4, pp. 83–93, Apr. 1988.
[98] L. Zadeh, "Fuzzy sets," *Inform. and Control*, vol. 8, no. 3, pp. 338–353, 1965.
[99] K. Anindita, "A new approach for task scheduling of cloud computing using fuzzy," *Int. J. Innovative Res. Comput. Sci. Technol.*, vol. 3, pp. 112–116, Mar. 2015.
[100] P. Haratian, F. Safi-Esfahani, L. Salimian, and A. Nabiollahi, "An adaptive and fuzzy resource management approach in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 4, pp. 907–920, Oct. 2019.
[101] S. Srinivas, S. Anupama, and K.-J. Suvendu, "Efficient load balancing in cloud computing using fuzzy logic," *IOSR J. Eng.*, vol. 2, pp. 65–71, Jul. 2012.
[102] X. Wang, Y. Wang, Z. Hao, and J. Du, "The research on resource scheduling based on fuzzy clustering in cloud computing," in *Proc. 8th Int. Conf. Intell. Comput. Technol. Autom.*, Jun. 2015, pp. 1025–1028.
[103] Ritu and S. Jain, "A trust model in cloud computing based on fuzzy logic," in *Proc. IEEE Int. Conf. Recent Trends Electronics, Inform. Commun. Technol.*, May 2016, pp. 47–52.
[104] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, and Cybern.*, vol. SMC-15, no. 1, pp. 116–132, Feb. 1985.
[105] Y. Tao, X. Wang, X. Xu, and Y. Chen, "Dynamic resource allocation algorithm for container-based service computing," in *Proc. IEEE 13th Int. Symp. Auton. Decentralized Syst.*, Mar. 2017, pp. 61–67.
[106] G. Li *et al.*, "Fuzzy theory based security service chaining for sustainable mobile-edge computing," *Mobile Inform. Syst.*, vol. 2017, pp. 1–13, 2017.
[107] C. Qu and R. Buyya, "A cloud trust evaluation system using hierarchical fuzzy inference system for service selection," in *Proc. IEEE 28th Int. Conf. Adv. Inform. Netw. Appl.*, May 2014, pp. 850–857.
[108] M. Zarina, A. M. Aminu, S. W. N. Wan Nur, M. A. Mohamed, and M. D. Mustafa, "A genetic algorithm for optimal job scheduling and load balancing in cloud computing," *Int. J. Eng. Technol.*, vol. 7, pp. 290–294, 2018.
[109] P. Kumar and A. Verma, "Scheduling using improved genetic algorithm in cloud computing for independent tasks," in *Proc. Int. Conf. Adv. Comput., Commun. Informat.*, 2012, pp. 137–142.
[110] K. Shaminder and V. Amandeep, "An efficient approach to genetic algorithm for task scheduling in cloud computing environment," *I. J. Inform. Technol. Comput. Sci.*, vol. 10, pp. 74–79, 2012.
[111] Y. Wang, C. Guo, and J. Yu, "Immune scheduling network based method for task scheduling in decentralized fog computing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–8, 2018.
[112] Y.-K. Lin and C. S. Chong, "Fast GA based project scheduling for computing resources allocation in a cloud manufacturing system," *J. Intell. Manuf.*, vol. 28, no. 5, pp. 1189–1201, Jun. 2017.
[113] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3317–3329, Dec. 2015.
[114] K. Nitesh and K. Jitender, "A computation offloading framework to optimize energy utilization in mobile cloud computing environment," *Int. J. Comput. Appl. Inform. Technol.*, vol. 5, pp. 61–69, 2014.
[115] M. Goudarzi, M. Zamani, and A. Toroghi Haghighat, "A genetic-based decision algorithm for multisite computation offloading in mobile cloud computing," *Int. J. Commun. Syst.*, vol. 30, no. 10, pp. e3241:1–e3241:17, 2016.
[116] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. Salinas Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2020.2994232.
[117] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "Energy efficient computation offloading for multi-access MEC enabled small cell networks," in *Proc. IEEE Int. Conf. Commun. Workshops*, May 2018, pp. 1–6.
[118] Z. Li and Q. Zhu, "Genetic algorithm-based optimization of offloading and resource allocation in mobile-edge computing," *Information*, vol. 11, no. 2, pp. 1–11, 2020.
[119] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, (Morgan Kaufmann Series in Data Management Systems Series), 3rd ed. San Mateo, CA, USA: Morgan Kaufmann, 2011.
[120] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.
[121] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Commun. Surv. Tut.*, vol. 20, no. 4, pp. 2595–2621, Oct.–Dec. 2018.
[122] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Mobile Netw. Appl.*, to be published, doi: 10.1007/s11036-018-1177-x.
[123] D. S. Rani and M. Pounambal, "Deep learning based dynamic task offloading in mobile cloudlet environments," *Evol. Intell.*, to be published, doi: 10.1007/s12065-019-00284-9.
[124] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.
[125] Y. Shoham, R. Powers, and T. Grenager, "Multi-agent reinforcement learning: A critical survey," Computer Science Department, Stanford University, Stanford, CA, USA, Tech. Rep., 2003.
[126] A. T. Nassar and Y. Yilmaz, "Reinforcement-learning-based resource allocation in fog radio access networks for various IoT environments," 2018, *arXiv:1806.04582*.
[127] L. Wang, P. Huang, K. Wang, G. Zhang, L. Zhang, N. Aslam, and K. Yang, "Rl-based user association and resource allocation for multi-UAV enabled MEC," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf.*, Jun. 2019, pp. 741–746.
[128] Z. Peng, D. Cui, Y. Ma, J. Xiong, B. Xu, and W. Lin, "A reinforcement learning-based mixed job scheduler scheme for cloud computing under SLA constraint," in *Proc. IEEE 3rd Int. Conf. Cyber Secur. Cloud Comput.*, Jun. 2016, pp. 142–147.
[129] X. Dutreilh, S. Kirgizov, O. Melekhova, J. Malenfant, N. Rivierre, and I. Truck, "Using reinforcement learning for autonomic resource allocation in clouds: Towards a fully automated workflow," in *Proc. 7th Int. Conf. Autonomic Auton. Syst.*, 2011, pp. 67–74.
[130] X. Liu, Z. Qin, and Y. Gao, "Resource allocation for edge computing in IoT networks via reinforcement learning," 2019, *arXiv:1903.01856*.
[131] T. Q. Dinh, Q. D. La, T. Q. S. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6353–6367, Dec. 2018.
[132] F. Youssef and B. Houda, "Deep reinforcement learning overview of the state of the art," *J. Autom., Mobile Robot. Intell. Syst.*, vol. 12, no. 3, pp. 20–39, 2018.
[133] A. Mosavi, P. Ghamisi, Y. Faghan, P. Duan, and S. Shamshirband, "Comprehensive review of deep reinforcement learning methods and applications in economics," 2020, *arXiv:202003.0309.v1*.
[134] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.
[135] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations Trends Mach. Learn.*, vol. 11, no. 3–4, pp. 219–354, 2018.
[136] J. Lin, Z. Peng, and D. Cui, "Deep reinforcement learning for multi-resource cloud job scheduling," in *Neural Information Processing*, L. Cheng, A. C. S. Leung, and S. Ozawa, Eds. Berlin, Germany: Springer International Publishing, 2018, pp. 289–302.
[137] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, and P. Li, "Energy-efficient scheduling for real-time systems based on deep Q-learning model," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 1, pp. 132–141, Jan.–Mar. 2019.
[138] T. Yang, Y. Hu, M. C. Gursoy, A. Schmeink, and R. Mathar, "Deep reinforcement learning based resource allocation in low latency edge computing networks," in *Proc. 15th Int. Symp. Wireless Commun. Syst.*, Aug. 2018, pp. 1–5.

[139] H. Meng, D. Chao, and Q. Guo, "Deep reinforcement learning based task offloading algorithm for mobile-edge computing systems," in *Proc. 4th Int. Conf. Math. Artif. Intell.*, 2019, pp. 90–94.

[140] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[141] J. Zhang, H. Guo, and J. Liu, "A reinforcement learning based task offloading scheme for vehicular edge computing network," in *Artificial Intelligence for Communications and Networks*, S. Han, L. Ye, and W. Meng, Eds. Berlin, Germany: Springer International Publishing, 2019, pp. 438–449.

[142] Z. Ning *et al.*"Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1060–1072, Dec. 2019.

[143] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digital Commun. Netw.*, vol. 5, no. 1, pp. 10–17, 2019.

[144] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11 158–11 168, Nov. 2019.

[145] Z. Ning, P. Dong, X. Wang, J. J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–24, Jan. 2019.

[146] A. Teske, R. Falcon, R. Abielmona, and E. Petriu, "Automating maritime risk assessment with genetic fuzzy systems," in *Proc. 2nd Int. Symp. Fuzzy Rough Sets*, 2017, pp. 1–10.

[147] S. Tzafestas and K. Zikidis, "NeuroFAST: on-line neuro-fuzzy ART-based structure and parameter learning TSK model," *IEEE Trans. Syst., Man Cybern., Part B (Cybern.)*, vol. 31, no. 5, pp. 797–802, Oct. 2001.

[148] A. M. Manasrah and H. B. Ali, "Workflow scheduling using hybrid GA-PSO algorithm in cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–16, Jul. 2018.

[149] C. Liu, C. Zou, and P. Wu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing," in *Proc. 13th Int. Symp. Distrib. Comput. Appl. Bus., Eng. Sci.*, Nov. 2014, pp. 68–72.

[150] S. Rashidi and S. Sharifian, "A hybrid heuristic queue based algorithm for task assignment in mobile cloud," *Future Generation Comput. Syst.*, vol. 68, pp. 331–345, 2017.

[151] N. Moganarangan, R. Babukarthik, S. Bhuvaneswari, M. S. Basha, and P. Dhavachelvan, "A novel algorithm for reducing energy-consumption in cloud computing environment: Web service computing approach," *J. King Saud Univ. - Comput. Inf Sci.*, vol. 28, no. 1, pp. 55–67, Jan. 2016.

[152] Z. Zhang and X. Zhang, "A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation," in *Proc. 2nd Int. Conf. Ind. Mechatronics Autom.*, May 2010, vol. 2, pp. 240–243.

[153] X. Wei, J. Fan, Z. Lu, and K. Din, "Application scheduling in mobile cloud computing with load balancing," *J. Appl. Math.*, vol. 2013, pp. 1–13, 2013.

[154] X. Liu, Z. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.

[155] W. Bao, H. Ji, X. Zhu, J. Wang, W. Xiao, and J. Wu, "ACO-based solution for computation offloading in mobile cloud computing," *Big Data Inform. Analytics*, vol. 1, pp. 1–13, Jan. 2016.

[156] A. Ragmani, A. El Omri, N. Abghour, K. Moussaid, and M. Rida, "An improved scheduling strategy in cloud computing using fuzzy logic," in *Proc. Int. Conf. Big Data Adv. Wireless Technol.*, 2016, pp. 22:1–22:9.

[157] S. F. Issawi, A. Al Halees, and M. Radi, "An efficient adaptive load balancing algorithm for cloud computing under bursty workloads," *Eng., Technol. Appl. Sci. Res.*, vol. 5, no. 3, pp. 795–800, 2015.

[158] H. S. Mondal, M. T. Hasan, T. K. Karmokar, and S. Sarker, "Improving quality of service in cloud computing architecture using fuzzy logic," in *Proc. 4th Int. Conf. Adv. Elect. Eng.*, Sep. 2017, pp. 149–152.

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

[160] M. S. Q. Zulkar Nine, M. A. K. Azad, S. Abdullah, and R. M. Rahman, "Fuzzy logic based dynamic load balancing in virtualized data centers," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2013, pp. 1–7.

[161] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE Int. Conf. Commun.*, May 2018, pp. 1–6.

[162] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerging Topics Comput.*, to be published, doi: 10.1109/TETC.2019.2902661.

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

[164] L. Quan, Z. Wang, and F. Ren, "A novel two-layered reinforcement learning for task offloading with tradeoff between physical machine utilization rate and delay," *Future Internet*, vol. 10, no. 7, pp. 1–17, 2018.

[165] S. Sundar and B. Liang, "Gaming and learning approaches for multi-user computation offloading," in *Proc. IEEE Veh. Technol. Conf.*, 2017, pp. 24–27.

[166] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile edge computation offloading using game theory and reinforcement learning," 2017, *arXiv:1711.09012*.

[167] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 64–69, May 2019.

[168] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," in *Proc. IEEE 88th Veh. Technol. Conf.*, 2018, pp. 1–6.

[169] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," 2018, *arXiv preprint arXiv:1812.07394*.

[170] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, and M. Guizani, "Security in mobile edge caching with reinforcement learning," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 116–122, Jun. 2018.

[171] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2018, pp. 1–6.

[172] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2019.2928811.

[173] S. Vimal, M. Khari, N. Dey, R. G. Crespo, and Y. H. Robinson, "Enhanced resource allocation in mobile edge computing using reinforcement learning based moaco algorithm for IIOT," *Comput. Commun.*, vol. 151, pp. 355–364, 2020.

[174] A. S. Saljoughi, M. Mehvarz, and H. Mirvaziri, "Attacks and intrusion detection in cloud computing using neural networks and particle swarm optimization algorithms," *Emerg. Sci. J.*, vol. 1, no. 4, pp. 179–191, Dec. 2017.

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

[176] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "Deep learning based joint resource scheduling algorithms for hybrid MEC networks," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6252–6265, Jul. 2020.

[177] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–6.

[178] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 160–168, 2019.

[179] R. K. Naha *et al.*, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47 980–48 009, 2018.

[180] Y. Zhang, B. Di, P. Wang, J. Lin, and L. Song, "HetMEC: Heterogeneous multi-layer mobile edge computing in the 6G era," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4388–4400, Apr. 2020.

[181] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Commun. Surv. Tut.*, vol. 20, no. 3, pp. 1826–1857, Jul.–Sep. 2018.

[182] Y. Yuan and F. Wang, "Blockchain and cryptocurrencies: Model, techniques, and applications," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 48, no. 9, pp. 1421–1428, Sep. 2018.

[183] K. R. Choo, Z. Yan, and W. Meng, "Blockchain in industrial IoT applications: Security and privacy advances, challenges, and opportunities," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4119–4121, Jun. 2020.

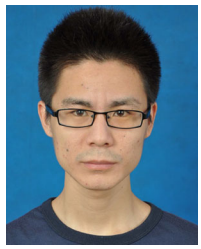
Muhammad Asim received the M.S. degree in Mathematics from University of Peshawar, Peshawar, Pakistan, in 2013 and the M.Phil. degree in Mathematics from the Kohat University of Science & Technology, Kohat, Pakistan, in 2016. He is currently pursuing the Ph.D. degree in Computer Science and Technology, Central South University, Changsha, China. His current research interests include computational intelligence, cloud computing, and edge computing.

Yong Wang (Senior Member, IEEE) received the Ph.D. degree in control science and engineering from Central South University, Changsha, China, in 2011.

He is a Professor with the School of Automation, Central South University, Changsha. His current research interests include the theory, algorithm design, and interdisciplinary applications of computational intelligence.

Dr. Wang is an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the *Swarm and Evolutionary Computation*. He was a recipient of Cheung Kong Young Scholar by the Ministry of Education, China, in 2018, and a Web of Science highly cited researcher in Computer Science in 2017 and 2018.

Kezhi Wang (Member, IEEE) received the B.E. and M.E. degrees from the School of Automation, Chongqing University, China, in 2008 and 2011, respectively. He received the Ph.D. degree in Engineering from the University of Warwick, U.K. in 2015. He was a Senior Research Officer in University of Essex, U.K. Currently he is a Senior Lecturer with Department of Computer and Information Sciences at Northumbria University, U.K. His research interests include wireless communications and machine learning.



Pei-Qiu Huang received the B.S. degree in automation and the M.S. degree in control theory and control engineering both from Northeastern University, Shenyang, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree in control science and engineering, Central South University, Changsha, China. His current research interests include evolutionary computation, bilevel optimization, and mobile edge computing.