Peer-to-Peer Networking and Applications (2022) 15:1029–1044
https://doi.org/10.1007/s12083-021-01273-5

Energy-effective artificial internet-of-things application deployment in edge-cloud systems

Zhengzhe Xiang[1] · Yuhang Zheng[1,3] · Mengzhu He[1] · Longxiang Shi[1] · Dongjing Wang[2] · Shuiguang Deng[1,3] · Zengwei Zheng[1]

Received: 30 July 2021 / Accepted: 11 November 2021 / Published online: 1 December 2021
© The Author(s) 2021

Abstract
Recently, the Internet-of-Things technique is believed to play an important role as the foundation of the coming Artificial Intelligence age for its capability to sense and collect real-time context information of the world, and the concept Artificial Intelligence of Things (AIoT) is developed to summarize this vision. However, in typical centralized architecture, the increasing of device links and massive data will bring huge congestion to the network, so that the latency brought by unstable and time-consuming long-distance network transmission limits its development. The multi-access edge computing (MEC) technique is now regarded as the key tool to solve this problem. By establishing a MEC-based AIoT service system at the edge of the network, the latency can be reduced with the help of corresponding AIoT services deployed on nearby edge servers. However, as the edge servers are resource-constrained and energy-intensive, we should be more careful in deploying the related AIoT services, especially when they can be composed to make complex applications. In this paper, we modeled complex AIoT applications using directed acyclic graphs (DAGs), and investigated the relationship between the AIoT application performance and the energy cost in the MEC-based service system by translating it into a multi-objective optimization problem, namely the CA^{3}D problem — the optimization problem was efficiently solved with the help of heuristic algorithm. Besides, with the actual simple or complex workflow data set like the Alibaba Cloud and the Montage project, we conducted comprehensive experiments to evaluate the results of our approach. The results showed that the proposed approach can effectively obtain balanced solutions, and the factors that may impact the results were also adequately explored.

Keywords Edge computing · Internet-of-things · Service deployment

1 Introduction

The rapid development and evolution of Artificial Intelligence (AI) theory and technology have brought a revolution to current information technology architectures. Especially, Internet-of-things (IoT) is one of them that faces both

This article is part of the Topical Collection: *Special Issue on Green Edge Computing*
Guest Editors: Zhiyong Yu, Liming Chen, Sumi Helal, and Zhiwen Yu

✉ Zengwei Zheng
zhengzw@zucc.edu.cn

Zhengzhe Xiang
xiangzz@zucc.edu.cn

Yuhang Zheng
zyhxxds_ludwig@zju.edu.cn

Mengzhu He
hemz@zucc.edu.cn

Longxiang Shi
shilx@zucc.edu.cn

Dongjing Wang
Dongjing.Wang@hdu.edu.cn

Shuiguang Deng
dengsg@zju.edu.cn

[1] Intelligent Plant Factory of Zhejiang Province Engineering Lab, Zhejiang University City College, Hangzhou, China

[2] Computer & Software School, Hangzhou Dianzi University, Hangzhou, China

[3] College of Computer Science and Technology, Zhejiang University, Hangzhou, China

challenges and opportunities because of its role as the data source of the real-world. The concept of Artificial Intelligence of Things (AIoT) is the combination of Artificial intelligence technologies with the Internet of things infrastructure to achieve more efficient IoT operations, improve human-machine interactions and enhance data management and analytic. According to the report of GSMA[1], the global total of cellular IoT connections is forecasted to reach 3.2 billion by 2024. There would be no doubt that the tremendous increasing connections will create a huge AIoT application market that draws the attention of the world. Based on the IoT technology, a reliable publish/subscribe interaction framework can be established between IoT devices and AIoT application developers so that high-quality data can be collected systematically. Traditionally, this collecting process is conducted with the end-cloud mode, the widely distributed but resource-constrained IoT devices only need to sense and upload the real world information to the cloud, and the cloud will handle the data processing. However, the latency brought by long-distance transmission and traffic congestion of huge data in the network, as well as the high cost like energy consumption brought by data pre-processing limits its wide application in the typical centralized architecture.

Fortunately, Multi-access Edge Computing (MEC) technique is proposed to solve the aforementioned problems [1–3]. Specifically, MEC is a novel paradigm that emerges recently as a reinforcement of mobile cloud computing, to optimize the mobile resource usage and enable wireless network to provide context-aware services [4, 5]. With the help of MEC, computation and transmission between mobile devices and the cloud are partly migrated to edge servers. Therefore, users can easily connect to their nearby edge servers via wireless network [6] and offload their tasks to them. The short-distance connection between users and edge servers can dramatically reduce the latency, and the computation capability of the edge servers is quite qualified to finish those conventional tasks. What's more, with the help of the container platforms in the limelight like `Kubernetes`, it will be easy to manage services (e.g. the data pre-processing services) in the MEC environment. However, these advantages cannot be the excuse of the carelessness in planning the multi-source AIoT sensing and analysing tasks — if the related services are not assigned to appropriate hosts, it may even obtain lower-quality result with much higher cost. More critically, as the edge servers are all resource-constrained [7, 8] and energy-consuming [9–12], there would be no enough resources for them to run if the data pre-processing services are not deployed on appropriate edge servers. Thus, it becomes more and more important to design a service deployment scheme as well as a resource allocation scheme to balance the quality and cost. The main contributions are summarized as follows:

1. We investigated the development of artificial intelligence of things technology and discussed the feasibility of adopting the multi-access edge computing architecture to optimize the performance of the AIoT systems.
2. We modeled the complex AIoT application with a directed acyclic graph, so that the execution of an AIoT application could be decomposed to several ordered AI services.
3. Based on the proposed application model, we constructed an appropriate metric to measure the AIoT application system, and mathematically modeled the service deployment problems which aimed to optimize the performance and the cost under the constraints edge resources as a multi-objective programming problem.
4. We designed and implemented an MOEA/D based algorithm to solve the problem, and conducted a series of experiments to evaluate the performance of the solutions. The results verified the improvement achieved by the proposed algorithm compared with other existing baselines. Besides, different configurations of the system were also investigated to explore the impacts of related factors.

The rest of this paper is organized as follows. Section 2 introduces how multi-access edge computing techniques can be used in optimizing AIoT applications with the example of a famous AI model. Section 3 shows some representative research works about service placement and resource allocation in MEC environment. Section 4 presents definitions, concepts and components of the problem to be solved. Section 5 describes the approaches we adopted to solve this problem. Section 6 shows the experimental results including the factors that affect our algorithms. Finally, Sect. 7 concludes our contribution and outlines future work.

2 Motivation scenario

Recently, AI research has become more and more structural and systemic with the prosperity of deep learning (DL) theory and tools recently. With the help of mature libraries like Tensorflow, PyTorch, MindSpore, etc., researchers and developers can easily build their own models like building blocks. One main factor that facilitates this popularization lies on the common structure of these deep learning models – the directed acyclic graph (DAG) based computation workflow. There are many existing examples in AI research exhibiting the DAG structures. For example, Fig. 1 shows the structure[2]

[1] https://www.gsmaintelligence.com/

[2] https://d2l.ai/chapter_recommender-systems/deepfm.html

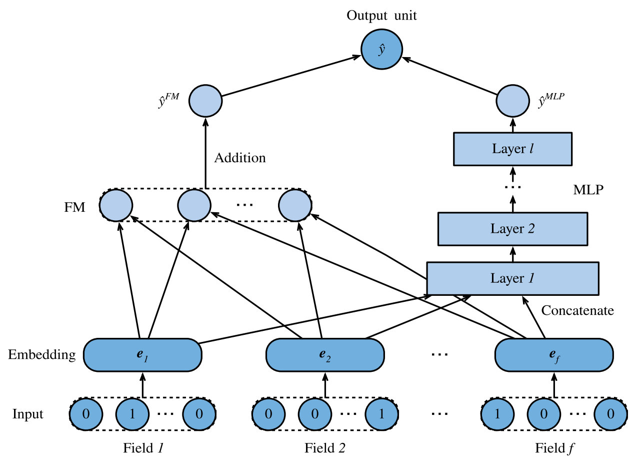

Fig. 1 The workflow of DeepFM

of DeepFM, a famous recommendation model proposed to predict click-through rate (CTR). Specifically, in this model, features of different fields are collected and wrapped as input, transformed to dense vectors with several embedding layers, and then are separately sent to the factorization machine (FM) and multi-layer perceptron (MLP) layers to generate the final output — we can clearly observe the data dependency and logic dependency in Fig. 1.

Generally, these DL-based AI models can be deployed on cloud servers with sufficient resources, and the data collected by IoT devices are uploaded to these servers together for further inference. However, long-distance communication between IoT devices and cloud servers may cause unavoidable delays. At the same time, there is no need to upload the context information collected in different regions to the cloud instead of processing it on-site. Therefore, if different components of an AI model are reasonably deployed using the multi-access edge computing architecture, the data transmission efficiency will be improved and the performance of AI tasks in the IoT environment will be greatly ameliorated.

3 Related work

3.1 Service placement in MEC

The issue of service placement is not a novel one, since how the services are placed will dramatically affect the performance of a parallelism and distributed system, especially when the definition of performance varies in different scenarios — the optimal placement strategies are usually derived according to the objectives that people mainly focus on. For example, Ouyang et al. addressed the service placement challenge in terms of the performance-cost trade-off [13]. They applied the Lyapunov optimization technique to study the edge service performance optimization problem under long-term cost budget constraint. Similarly, Pasteris et al. considered the heterogeneity of edge node characteristics and user locations in optimizing the performance of MEC by placing multiple services [14]. They partitioned each edge node into multiple slots, where each slot contains one service, and proposed a deterministic approximation algorithm to solve it after reducing the problem to a set cover problem. Roy et al. went further on the similar topic by introducing the users' path prediction model in such a scenario [15]. They formulated the service replica placement problem as a multi-objective integer linear programming problem, and used binary particle swarm optimization algorithm to achieve near-optimal solutions within polynomial time. Yuan et al. used a greedy approximation algorithm to solve the service placement problem under the constraints of computing and storage resources [16]. They also adopted a 2-time-scale framework to reduce the higher operating costs caused by frequent cross cloud service migration. To achieve dynamic service placement, based on Lyapunov optimization method, Ning et al. proposed an approximation-based stochastic algorithm to approximate the expected future system utility, then a distributed Markov approximation algorithm is used to determine the service configuration [17]. Han et al. focused on the online multi-component service placement in edge cloud networks [18]. Considering the dependency between service components, they analyzed the delay of tree-like services solved the problem by an improved ant colony algorithm.

3.2 Resource allocation in MEC

The resource allocation issue follows after deciding the appropriate edge server to place service instances. The resource allocation issue is important and it is widely discussed in the research of communication and distributed system, especially in the research of computation offloading, the key problem of the MEC paradigm. For example, Yu et al. considered a cloudlet that provides services for multiple mobile devices [19], and they proposed a joint scheduling algorithm that guided the sub-carrier allocation for Orthogonal Frequency-Division Multiplexing Access (OFDMA) system and CPU time allocation for the cloudlet. Wang et al. also tried to explore the relationship between cost and resource. They formulated computation offloading decision, resource allocation and content caching strategy as an optimization problem, considering the total revenue of the network [20]. Focusing on saving energy of mobile users, Shuwaili et al. proposed a resource allocation approach over both communication and computation resources, while You et al. [21] also considered the resource of the cloud. Guo et al. took the average packet delay as the optimization goal of the edge container resource allocation problem

[22], and proposed a delay-sensitive resource allocation algorithm based on A3C (asynchronous advantage actor-critic) to solve it. Bahreini et al. expressed the edge resource allocation problem (ERP) as a mixed integer linear problem (MILP) [23], proposed an auction-based mechanism, and proved that the proposed mechanism is individually rational, resulting in non-jealous allocation. It solves resource allocation and monetization challenges in MEC system. Yang et al. studied joint computing partition and resource allocation for delay-sensitive applications in MEC system [24]. They proposed a new efficient off-line algorithm, namely multi-dimensional Search adjustment Algorithm (MDSA), to solve this problem. In addition, they designed an online method, Cooperative Online Scheduling (COS), which is easy to deploy in real systems.

In summary, these researches are quite valuable because they shed light on the fundamental concepts and inspired the thoughts of related topics in application deployment in MEC environment. However, the relationship among service placement, resource allocation, application performance and the energy consumption is still under the sea. Therefore, we go further by combining the resource allocation and service placement problems together to explore the trade-off between application performance and the energy consumption based on these works, and apply a simple but effective heuristic approach that optimizes the system in the end (See Table 1).

Table 1 Symbol Description

Symbols	The physical meaning of the notations
H	the set of edge server, $\|H\| = n$
h_j	the j-th server in H
U_j	the set of IoT devices in the serving area of h_j
v_j^e	the average transmission rate between h_j and devices in U_j
v_j^c	the average transmission rate between h_j and the cloud
$b_{j,k}$	the average transmission rate between h_j and h_k
$\mu_j^\star$	the available computing resource of h_j
$S^{\mathbb{R}}$	the service set of the ECC system, $\|S^{\mathbb{R}}\| = m$
s_i	the i-th service in $\mathcal{S}$
$S^{\mathbb{V}}$	the virtual services that collect context-aware data around different edge servers, $\|S^{\mathbb{V}}\| = n$
$S = S^{\mathbb{R}} \cup S^{\mathbb{V}}$	the set of real services and virtual services
c_i	the i-th service in $S^{\mathbb{V}}$
I_i	the average input data size of s_i
O_i	the average output data size of s_i
w_i	the average workload of s_i
$\mu_{j,i}^k$	the resource that h_j allocates to s_i
$\boldsymbol{G}$	the AIoT application set of the system, $\|\boldsymbol{G}\| = K$
$G_k = (S_k, E_k)$	the k-th AIoT application in $\|\boldsymbol{G}\|$
p_i^k	the edge server index where the service s_i in AIoT application G_k is placed on
$\mathcal{F}_k(s_i)$	the precursor set of service s_i in AIoT application G_k
η_j	the energy conversion rate of h_j

4 System model and problem description

Although the example in Sect. 2 has given a brief illustration about the scenario, more details like costs, capacities and the cases of multi-application are ignored for briefness. Therefore, we will give a complete system model and then describe the performance-cost optimization problem.

4.1 Server and network

In a typical AIoT system, the remote server or cloud server is responsible for processing all the IoT context information sensed by IoT devices distributed in specific areas. However, it will be much different when introducing the edge-cloud system. In an edge-cloud system, a set of edge servers $H = \{h_1, h_2, ..., h_n\}$ will be located to collect n different types of context data in these specific sensing areas, while each of the edge server is equipped with cloud-like computing and storage capability. The edge servers can easily extract the useful information from received data and perform analysis with their resources. In general, it is the mobile base station that acts the role of edge server [25]. To make full use of the resources of these edge servers, they further make up an edge-side ad-hoc computing cluster. For every edge server $h_j \in H$, it can receive the information collected by IoT devices (the set of these devices is denoted with U_j) around, and the average transmission rate between edge server h_j and IoT devices in U_j is v_j^e. Meanwhile, if it is necessary, data may be routed to and processed by any anther reachable edge server via the connection between edge servers. Formally, we use $b_{j,k}$ to describe the average transmission rate between the j-th edge server (*source*) and the k-th one (*target*). Since all edge servers can communicate with the cloud in an edge-cloud system, we use v_j^c to denote the average transmission rate between the cloud server and the j-th edge server. Particularly, we set $b_{j,k} = v_j^e$ if the *source* is U_j and set $b_{j,k} = v_j^c$ if the *target* is the cloud for simplification. The computing resource available on server h_j is described as $\mu_j^\star$, which means the workloads (e.g., data size in bit) the server can handle on average within one second (bps). Without loss of generality, here we just consider the computation resource like CPU because most data processing tasks are computation-sensitive and the storage resource is adequate. The researchers can easily extend it by introducing more kinds of resources and their corresponding estimation models.

4.2 DAG-based AIoT application

Edge servers use program modules with specific functionalities to finish data processing tasks, and these program modules are usually called *services*. A service can be launched as an instance with the help of popular PaaS technology like Kubernetes. Here we assume $S^{\mathbb{R}} = \{s_1, s_2, ..., s_m\}$ is the set of services that are involved in the edge-cloud system for information processing, and assume $S^{\mathbb{V}} = (c_1, c_2, ..., c_n)$ is the virtual service set which stands for the collection of context data on the IoT devices in different regions. Evidently, these virtual services are closely bounded with the edge servers. For example, s^v_z should be deployed on edge server h_z. For every $s_i \in S^{\mathbb{R}} \cup S^{\mathbb{V}}$, we use I_i to describe the average size of data received by s_i, O_i to describe the average size of data generated by s_i, and w_i to describe the average workload of processing the received data. Apparently, I_i and w_i will be zero when $s_i \in S^{\mathbb{V}}$ because we treat the IoT devices as data generator here.

However, these atomic services cannot handle the scenarios individually where requirements are complex. Therefore, people develop service composition technology by putting them together and invoke them in a certain order. Generally, we can use $G = (S', E)$, a directed acyclic graph (DAG) to describe an AIoT application with its business logic by revealing the execution order of its related services. Here $S' \subset S^{\mathbb{R}} \cup S^{\mathbb{V}}$ is the related service set, and $E = \{< s_i \rightarrow s_j > \mid s_i, s_j \in S'\}$ is the set of edges. By using the services in G according to the vertex topological order, and treating the output of s_i as the input of s_j for all $s_i \rightarrow s_j \in E$ as a relay race, the AIoT application denoted with G can be executed step by step. Obviously, for any two individual services $s_i, s_j \in S'$, the output of s_i will be the input of s_j if $s_i \rightarrow s_j \in E$, and we can approximately assume that $I_j = O_i$ in this case.

4.3 AIoT application deployment scheme

Obviously, there will be more than one AIoT application in an edge-cloud system. If we assume there are K AIoT applications $\boldsymbol{G} = (G_1, G_2, ..., G_K)$ in the system, and $G_k = (S_k, E_k)$ stands for the k-th AIoT application which uses several services in $S_k = S^{\mathbb{V}}_k \cup S^{\mathbb{R}}_k$ (the involved virtual and real services of the k-th AIoT application), we should consider how these applications can be deployed next.

Usually, given an arbitrary AIoT application $G_k=(S_k, E_k)$ in $\boldsymbol{G}$, we use a placement vector $\boldsymbol{p}^k=\{p^k_i\}_{i=1}^{|S^{\mathbb{R}}_k|}$ (it's not necessary to consider the placement of service in $S^{\mathbb{V}}$ because they are context-aware and bounded with the edge servers), and a resource allocation matrix $\boldsymbol{\mu}^k=\{\mu^k_{j,i}\}_{j=1,i=1}^{n,|S^{\mathbb{R}}_k|}$ to describe the *deployment scheme* of the k-th AIoT application, where $p_i \in [1, n]$ is the index of the selected edge server to deploy service s_i and $\mu_{j,i}$ is the resource allocated to service s_i on edge server h_j. As the selected edge server belongs to H, and the used resource cannot exceed the maximum capacity, we will have the following constraints $\forall s_i \in S^{\mathbb{R}}_k$:

$$\begin{aligned} & 1 \leq p^k_i \leq n \\ & \sum_{k=1}^{K}\sum_{i=1}^{m} \mu^k_{j,i} \leq \mu^\star_j \end{aligned} \tag{1}$$

To demonstrate the concepts above, here use a system with 3 AIoT applications in Fig. 2 to help understanding. In the

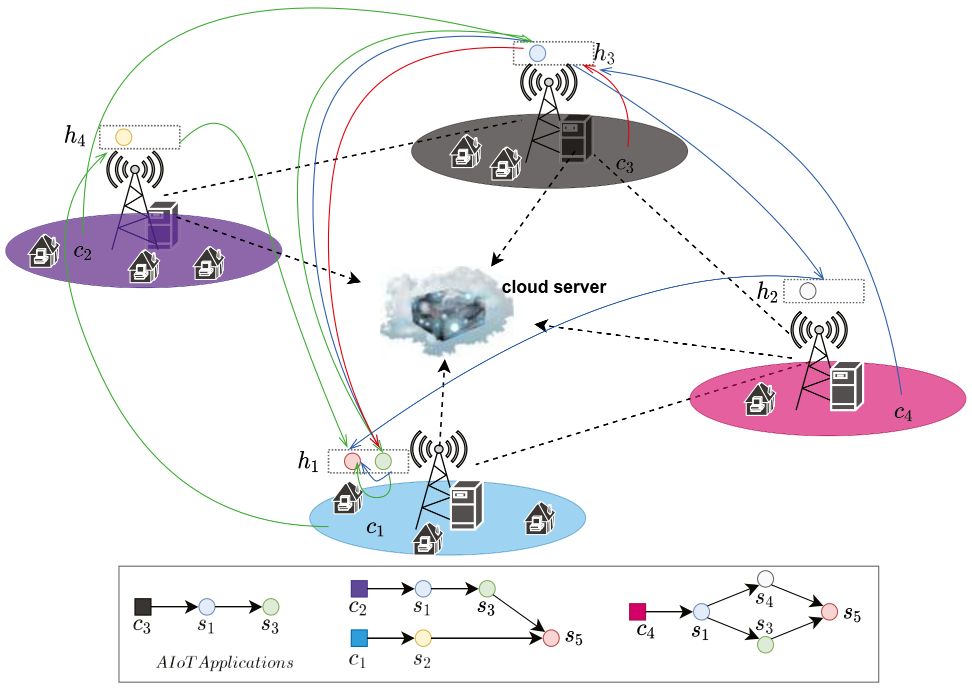

Fig. 2 An example of deploying 3 AIoT applications

example shown in Fig. 2, we can find that there are 4 edge servers which cooperate with each other and connect to the cloud making up an edge-cloud system. Particularly, as these 4 edge servers locate in different places and serve different users, the collected data will have different contexts (shown in c_1, c_2, c_3, c_4). To make full use of the collected context-aware data, there are 3 AIoT applications $G_1 = (\{c_1,s_1,s_3\}, \{c_1 \rightarrow s_1, s_1 \rightarrow s_3\})$, $G_2 = (\{c_1,c_2,s_1, s_2, s_3, s_5\},\{c_1 \rightarrow s_2, c_2 \rightarrow s_1, s_1 \rightarrow s_3,s_3 \rightarrow s_5,s_2 \rightarrow s_5\})$ and $G_3 = (\{c_4,s_1,s_3,s_4,s_5\},\{c_4 \rightarrow s_1,s_1 \rightarrow s_4,s_1 \rightarrow s_3,s_4 \rightarrow s_5,s_3 \rightarrow s_5\})$ listed in the box, which stand for 3 different DL-based AI models in with specific data source (e.g. the first AIoT G_1 receives the data c_3 from h_3) in DAGs, to complete the data analysis tasks. Typically, these AIoT applications are deployed on the cloud, so all the context-aware data collected by IoT devices will be processed after being uploaded to the cloud. However, in the MEC paradigm the services involved in these AIoT applications can be separately deployed on the edge servers. Therefore, we can find that services s_1-s_5 (in colored circles) are deployed in edge server h_1-h_4. In this scenario, the collected context-aware data c_3 will firstly processed by s_1 on h_3 and then by s_3 on h_1 (shown with the red curve) to implement the function of G_1.

4.4 AIoT application performance evaluation

AIoT applications will keep running on the edge servers to sense the world by collecting the configuration of the physical world in the service systems. Hence, it is vital for the AIoT application developers to improve the performance of their applications, and the average time cost of the applications in the system will be a representative indicator to measure the system performance. Taking advantage of the dependency in DAGs by adding a dummy service $s_k^{\#}$ to G_k so that all the end services in G_k with 0 out-degree are directed to it, the completion time of $s_i \in S_k^{\mathbb{R}}$ can be represented as:

$$T^C(G_k, s_i) = \frac{w_i}{\mu^k_{p_i,i}} + \max\{T^C(G_k, s_z) + \frac{O_z}{b_{p_z,p_i}} | s_z \in \mathcal{F}(s_i)\} \tag{2}$$

and completion time of $s_i \in S_k^{\mathbb{V}}$ will be $T^C(s_i) = 0$, where $\mathcal{F}_k(s_i)$ is the precursor set of service s_i in AIoT application G_k. For example, given an AIoT application G_1 which is composed with three AIoT services in sequential order as $\{s_1 \rightarrow s_2 \rightarrow s_3\}$. We assume that service s_1 is currently deployed on edge server h_3, while s_2 on h_2 and s_3 on h_1. In this case, the used data will be collected by sensor c_2 in h_2's serving area. Therefore, we can have $S^{\mathbb{V}} = \{c_2\}$, $S^{\mathbb{R}} = \{s_1, s_2, s_3\}$, $E = \{c_1 \rightarrow s_1, s_1 \rightarrow s_2, s_2 \rightarrow s_3\}$, $\mathcal{F}_1(s_1) = \{c_2\}$, $\mathcal{F}_1(s_2) = \{s_1\}$, $\mathcal{F}_1(s_3) = \{s_2\}$. To calculate the total time cost of running G_1, we need to calculate $T^c(G_1, s_1)$ first:

$$T^c(G_1, s_1) = \frac{w_1}{\mu_{3,1}} + T^c(c_2) + \frac{I_1}{v_3^e} \tag{3}$$

As the c_2 is a sensor to collect data, it is obvious that $T^c(c_2) = 0$. Meanwhile, the s_1 is the closest service to input end, so the input data is I_1, which is equal to the output data of sensor c_2, namely O_{c_2}. Next, we start to calculate $T^c(G_1, s_2)$. According to recursive expression in Eq. (2), we will have

$$T^c(G_1, s_2) = \frac{w_2}{\mu_{2,2}} + T^c(G_1, s_1) + \frac{O_1}{b_{3,2}} \tag{4}$$

Because s_2 only has one precursor s_1, only $T^c(G_1, s_1) + \frac{O_1}{b_{3,2}}$ is used in this case. But if there are more precursor nodes, then we should choose the one takes the most time because it will be the bottleneck. Similarly,

$$T^c(G_1, s_3) = \frac{w_3}{\mu_{1,3}} + T^c(G_1, s_2) + \frac{O_2}{b_{2,1}} \tag{5}$$

Finally, the output data obtained by s_3 is transmitted to the cloud, so we have

$$T^c(G_1, s_{\#}) = T^c(G_1, s_3) + \frac{O_3}{v_1^c} \tag{6}$$

In this way, we can use the value of $T^C(G_k, s_k^{\#})$ to evaluate the time cost of the k-th AIoT application. Based on it, if the collected IoT context data packages are uploaded and used by the AIoT application with the frequency f_k, the average time cost of the applications in this edge-cloud system will be represented as:

$$T(\boldsymbol{G}) = \sum_{k=1}^{K} f_k \cdot T^C(G_k, s_k^{\#}). \tag{7}$$

4.5 Energy consumption model

It can be found that the driving force of the multi-access edge computing paradigm lies in its widely distributed, large-scale available edge resources (in order to complete as many tasks as possible locally). However, this feature will also result in the consumption of a large amount of energy when maintaining these edge servers. For example, in a typical multi-access edge computing scenario with base stations as edge servers, the power consumption of a single edge server will reach 2.2~3.7×10^3W. Considering the about 9.3 million base stations in China, the total power consumption may be as high as 2.046~3.441×10^{10}W. High energy consumption has brought great challenges to the promotion and popularization of the MEC paradigm. Therefore, here we also consider the energy consumption of running AIoT applications.

Generally, the major energy is consumed in the process of computing. The computation energy is influenced by the clock frequency of the chip, and some techniques like the dynamic voltage scaling (DVS) technology [26] can use this property to adaptively adjust the energy consumption. In CMOS circuits [27], the energy consumption is proportional to the supply voltage. Moreover, it has been observed that the clock frequency of the chip f is approximately linearly proportional to the voltage. Therefore, the energy consumption can be expressed as $E \propto f^2$ [28]. At the same time, as f is proportional to the allocated resource, we can model the energy consumption expense $C(\boldsymbol{G})$ of running applications:

$$C(\boldsymbol{G}) = \sum_{k=1}^{K} \sum_{j=1}^{n} \sum_{i=1}^{m} \eta_j w_i (\mu_{j,i}^k)^2. \tag{8}$$

4.6 Problem definition and formulation

Based on the introduction of related concepts, now we can give the definition of the **c**ontext-**a**ware **A**IoT **a**pplication **d**eployment (CA^{3}D) problem clearly. In this CA^{3}D problem, the AIoT application developers would like to have an appropriate deployment scheme so that the average time cost of their applications can be minimized, as well as the energy consumption expense in an given MEC-based architecture. Therefore, we can now formulate the CA^{3}D problem as follows:

$$\begin{aligned} P: \quad & \min_{\boldsymbol{x}=(\boldsymbol{p},\boldsymbol{\mu})} \; F(\boldsymbol{x}) = \Big(T(\boldsymbol{G}), C(\boldsymbol{G})\Big) \\ s.t. \quad & 1 \le p_i^k \le n \\ & \sum_{k=1}^{K} \sum_{i=1}^{m} \mu_{j,i}^k \le \mu_j^\star \end{aligned} \tag{9}$$

5 Approach

It is not hard to find that the objectives depend on the value of decision variables $\boldsymbol{p}$ and $\boldsymbol{\mu}$, and the bounded integer constraint for p_i^k makes the optimization problem mix-integer and nonlinear. Meanwhile, the requirement of optimizing both the application time cost and deployment cost (energy consumption) makes the problem to be a multi-objective optimization problem (MOOP). These properties challenge the solving of our CA^{3}D problem. Therefore, we turn to the heuristic method like MOEA/D [29] and try to find some sub-optimal solutions.

Typically, an MOOP is solved with a decomposition strategy, which decomposes the original problem into several scalar optimization sub-problems and optimizes them simultaneously [30, 31]. For example in the classic MOEA/D method, the Tchebycheff decomposition is used to measure the maximum weighted distance between the objectives and their minimums $\boldsymbol{z}^* = (z_T^*, z_C^*)$:

$$g^{te}(\boldsymbol{x}|\lambda, z^*) = \max\{\lambda_T |T(\boldsymbol{x}) - z_T^*|, \lambda_C |C(\boldsymbol{x}) - z_C^*|\} \tag{10}$$

, where $0 \le \lambda_C, \lambda_T \le 1$ and $\lambda_C + \lambda_T = 1$ are the constraints for weight vector $\lambda = (\lambda_T, \lambda_C)$. Obviously, the shorter the distance between $f(\boldsymbol{x})$ and its minimum is, the closer will the $\boldsymbol{x}$ be with the optimal solution. And with the weight vector, we can finally search the Pareto optimum in an iterative way. Algorithm 1 shows the detailed operations in solving the problem with the MOEA/D. In this process, each sub-problem will be optimized by using information from its several neighbors.

It can be found in Algorithm 1 that several evolutionary operators like *crossover* and *mutate* are involved. Actually, MOEA/D provides the possibility to use traditional evolutionary algorithms like Genetic algorithm (GA) [32] to solve multi-objective problems. Therefore, we borrow the operators in GA, a kind of meta-heuristic algorithm inspired by the process of natural selection, to solve our target problem. Therefore, after encoding the decision variable $\boldsymbol{p}$ and $\boldsymbol{\mu}$ as $\boldsymbol{p} = (p_1^1, ..., p_m^1, ..., p_1^K, ..., p_m^K)$ and $\boldsymbol{\mu} = (\mu_{1,1}^1, ..., \mu_{1,m}^1, ..., \mu_{n,1}^1, ..., \mu_{n,m}^1, ..., \mu_{1,1}^K, ..., \mu_{1,m}^K, ..., \mu_{n,1}^K, ..., \mu_{n,m}^K)$ and combining them to get $\boldsymbol{x}$, the genetic algorithm will be embedded to Algorithm 1 with these operators. Obviously, the *crossover* operator makes it possible to obtain better solutions, and the *mutation* operation gives the algorithm the ability to avoid premature convergence.

For the sake of simplicity and variable-controlling, we adopt the same parameter configuration for the following evolutionary algorithms including MOEA/D, that is, the initial population is $N = 200$, the number of iterations is $MAX\_ITER = 200$, the mutation probability is $p_m = 0.1$ and the crossover probability is $p_c = 0.8$. Meanwhile, the initial population is generated randomly. In the pseudo-code of the algorithm, we can clearly see that the main complexity of the MOEA/D algorithm comes from the *for* loop in lines 22-32, that is, to update their neighboring solutions for each individual where the number of individuals is N, the number of adjacent solutions is N_A. For each objective function, the same operation needs to be performed. And there are two objective functions in our algorithm, namely $T(\cdot)$ and $C(\cdot)$. Thus, we can obtain that in an evolutionary iteration, the time complexity of the algorithm is $O(N * N_A)$, and the overall time complexity is $O(MAX\_ITER * N * N_A)$.

Algorithm 1 MOEA/D Framework for the CA^{3}D problem

```
Require: $N$, $MAX\_ITER$, $N_A$
 1: for each $i \in [1, N]$ do
 2:     for each $j \in [1, N]$ do
 3:         $D_{i,j} = |\boldsymbol{w}^i - \boldsymbol{w}^j|$
 4:     end for
 5:     $D^K_{i,*}, D^V_{i,*} = findNeighborKeyValuePairs(D_{i,*}, N_A)$
 6: end for
 7: initialize population $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^N\}$
 8: V = $\{F(\boldsymbol{x}^1), F(\boldsymbol{x}^2), ..., F(\boldsymbol{x}^N)\}$
 9: initialize $z = (z_1, z_2)^T$
10: $count = 0$
11: while $count < MAX\_ITER$ do
12:     for each $i \in [1, N]$ do
13:         $\boldsymbol{x}^k, \boldsymbol{x}^l = randomSelect(D^K_{i,*})$
14:         $\boldsymbol{x}' = crossover(\boldsymbol{x}^k, \boldsymbol{x}^l, p_c)$
15:         $\boldsymbol{y}' = mutate(\boldsymbol{x}', p_m)$
16:         $\boldsymbol{y}' = heal(\boldsymbol{y}')$
17:         if $z_1 < T(\boldsymbol{y}')$ then
18:             $z_1 = T(\boldsymbol{y}')$
19:         end if
20:         if $z_2 < C(\boldsymbol{y}')$ then
21:             $z_2 = C(\boldsymbol{y}')$
22:         end if
23:         for each $j \in D^K_{i,*}$ do
24:             if $g^{te}(\boldsymbol{y}'|\boldsymbol{\lambda}^j, z) \leq g^{te}(\boldsymbol{x}^j|\boldsymbol{\lambda}^j, z)$ then
25:                 $\boldsymbol{x}^j = \boldsymbol{y}'$
26:                 $V^j = F(\boldsymbol{y}')$
27:             end if
28:             $EP' = \{y \mid (T(\boldsymbol{y}), C(\boldsymbol{y})) > F(\boldsymbol{y}')\}$
29:             $EP = EP$ - $EP'$
30:             $EP^d = \{$y$ \mid (T(\boldsymbol{y}), C(\boldsymbol{y})) < F(\boldsymbol{y}')\}$
31:             if $EP^d == \varnothing$ then
32:                 $EP = EP \cup \{\boldsymbol{y}'\}$
33:             end if
34:         end for
35:     end for
36:     $count = count + 1$
37: end while
```

6 Experiments and analysis

To fully explore the impacts of the solution derived from the MOEA/D based algorithm, we partially use the dataset of Alibaba Cluster data[3], which is published by Alibaba Group. It contains cluster trace of real production, and several containers are composed in DAG to finish complex tasks. Besides this, we also generate our experimental synthetic data with settings shown in Table 2 to perform our evaluations. What's more, to make the result convincing, the network and service parameters in this table are set close to reality. Besides the comparison with baselines, a series of comprehensive experiments were conducted on the simulation data in this section to explore the impact of different factors.

Meanwhile, to the best of our knowledge, the CA^{3}D problem is the first attempt to consider the deployment of AIoT services as well as optimizing the resource allocation strategy in the MEC environment, none of the existing approaches in former research works can be directly adopted in our problem. Thus, we select the following intuitive and representative strategies as baselines:

1. **Equality-sensitive Deployment (ESD)**. In the equality-sensitive deployment strategy, resources of edge servers will be allocated in an equal way to the service instance of all the application $G_k \in G$. This strategy is simple but easy to implement. It is practical in many cases so that it is used on plenty of real-world distributed systems.

Table 2 System Configurations

Param	Value	Param	Value
(m, n)	(9, 10)	K	8
d_i^O	$\mathcal{N}(5,1^2)$ MB	v_j^e	$\mathcal{N}(20,10^2)$ MB/s
w_i	$\mathcal{N}(20,10^2)\times10^4$ MI	$B_{j,k}$	$\mathcal{N}(20,10^2)$ kMB/s
η_j	$\mathcal{N}(10,1^2)$	μ_j	$\mathcal{N}(2,0.2^2)\times10^4$ MIPS

2. **Frequency-based Deployment (FBD)**. In the frequency-based deployment strategy, service instances of AIoT applications will be placed on the most frequent edge servers where the related services are mainly used in these areas. Meanwhile, the resources of edge servers will be allocated according to the frequency so that the most frequently used services will have the most resource. It is an unbalanced but useful strategy, it addresses the on-premise property of the MEC paradigm.
3. **Workload-aware Deployment (WAD)**. In the workload aware deployment strategy, service instances will be placed on the busy edge servers, and resources of edge servers will be allocated according to the workload of services, so that the heaviest services will have the most resource. WAD is a reinforcement of the FBD strategy because it distinguishes the burden of different requests.
4. **Transmission-aware Deployment (TAD)**. In the transmission aware strategy, resources of edge servers will be allocated according to the communication service placement preference because the transmission time cost is usually the major part that affects the performance.

With these settings in Table 2, we illustrate the average expense and service response time for these approaches in Fig. 3, and their running times in Fig. 4.

As we can see in Fig. 3, MOEA/D's optimization of the objective function shows its excellent capability to achieve a good balance between the performance and consumption. Different from the Pareto curve generated by the proposed algorithm, the results of baselines are scattered in this figure: among the baselines, TAD is significantly better than the other three strategies in terms of performance optimization, but the corresponding cost is also much higher. This is because the communication quality is often the main factor that affects the time cost. On the other hand, the expense in TAD will also increase when the resources of the server with better communication quality are all allocated. ESD is close to one solution of ours in the optimization of multi-objectives, but there is still a small gap. This is because evenly distributed resources among servers can play a positive role in the control of deployment expense — when there are not many requests, ESD also brings a splendid load balancing effect. However, our method can optimize the

[3] https://github.com/alibaba/clusterdata/

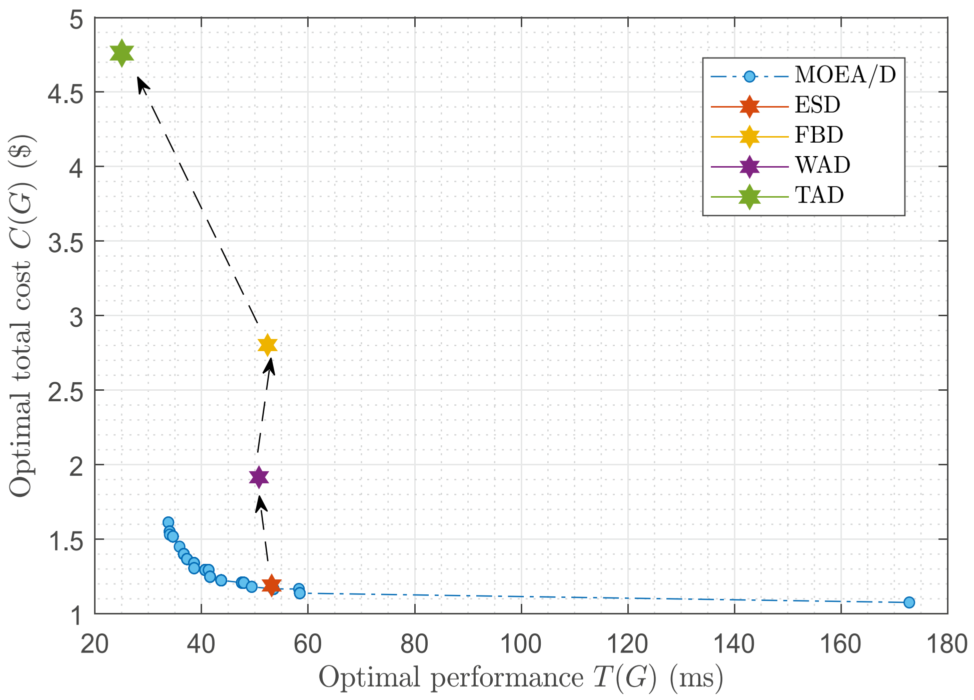

Fig. 3 The comparison with baselines

deployment of services and the resource allocation of each server in a more fine-grained manner so that it can achieve better results than ESD. In addition, our method can select the parameter configuration on the Pareto optimal curve according to different scenarios, and all the optimal solutions of the curve can achieve the best effect while ($T(\boldsymbol{G})$, $C(\boldsymbol{G})$) is balanced. Contrary to its good performance, the MOEA/D takes time to calculate the value of the decision variables. The running time under different problem scales (namely, the number of services, number of edge servers and number of IoT applications) are shown in the Fig. 4. As the problem scale increases, the running time also increases. But from the convergence of our method illustrated in Fig. 5 (the GEN in Fig. 5 means the evolution generation of the population.For example, the curve labeled with GEN=115 means this is the curve that shows the result of the algorithm after the 115-*th* iteration), we can find that the early-stop trick will be capable here as the curve after 179-*th* generation is almost approximate to that after 195-*th*, while the result Pareto curve is gradually moving to the direction of better performance with increase of generations. The comparisons above show the difference between our approach and other heterogeneous approaches. In these comparisons, some of the baselines are representative but not designed to solve this specific problem. Thus, as MOEA/D is one of the evolutionary algorithms, the other kinds of algorithms are also applied on this CA^{3}D problem to check whether it is appropriate to select the MOEA/D in solving this problem. The comparisons between these evolutionary algorithms are shown in Fig. 6. From Fig. 6 we can find that these algorithms approximately show the same capability in balancing the system performance and cost (except the MOEAD-DE algorithm) while the MOEA/D algorithm shows small advantages on the Pareto frontier.

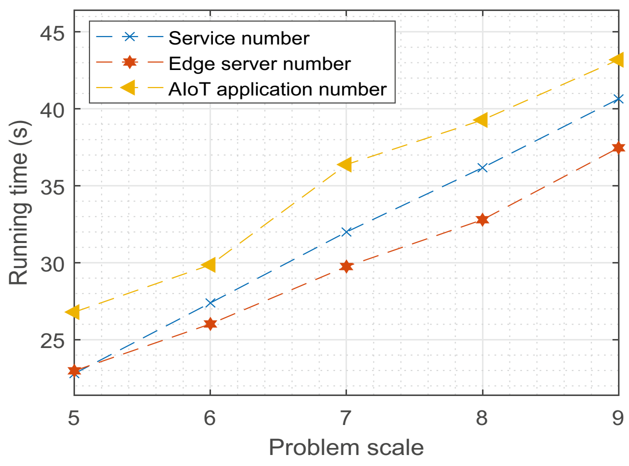

Fig. 4 The running times of different problem scales

For multi-objective optimization, one of the widely used indicators to measure the performance of the algorithm is HV (hyper-volume), which represents the hypercube formed by the individual in the solution set and the reference point in the target space volume. Hyper-volume can simultaneously evaluate the convergence and distribution of the solution set, which means the larger the HV value is, the better the overall performance of the algorithm will be. Based on the same data and hyper-parameter configuration, we run our algorithm 200 times

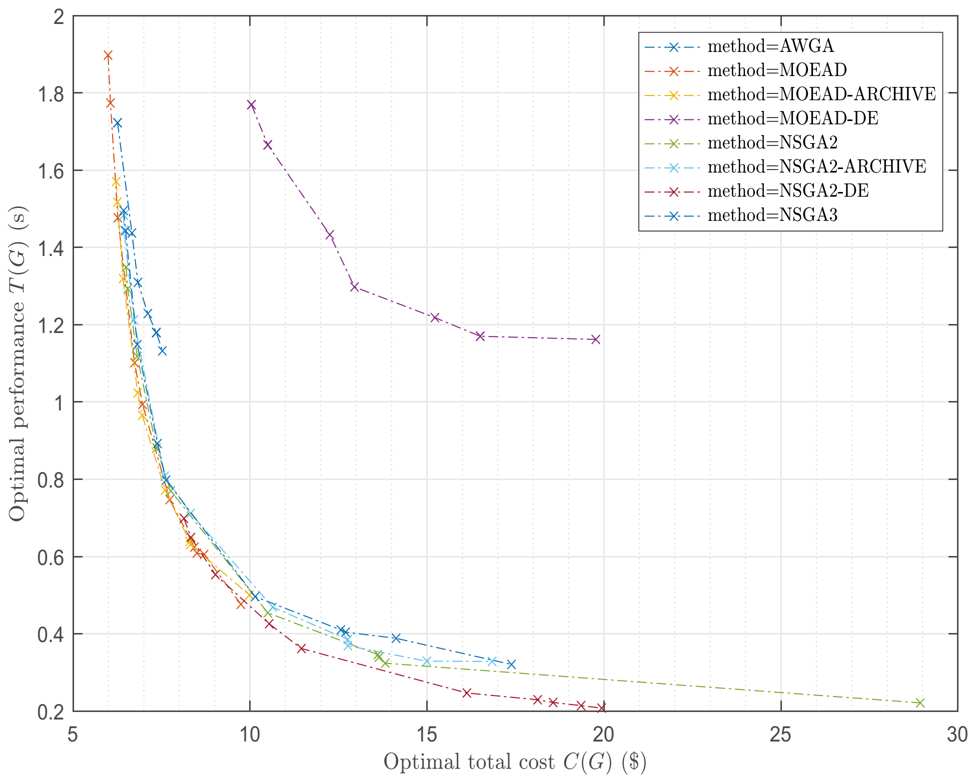

Fig. 6 The comparison with other evolutionary algorithms

and finally get the histogram of its distribution as shown in Fig. 7. Besides the histogram, the violin-plot is also demonstrated on the upper left corner of this figure to provide a clearer visualization. It can be seen that almost 80% of the HV data are above 0.70, indicating that our algorithm has good convergence and can get a nice result at the most of time.

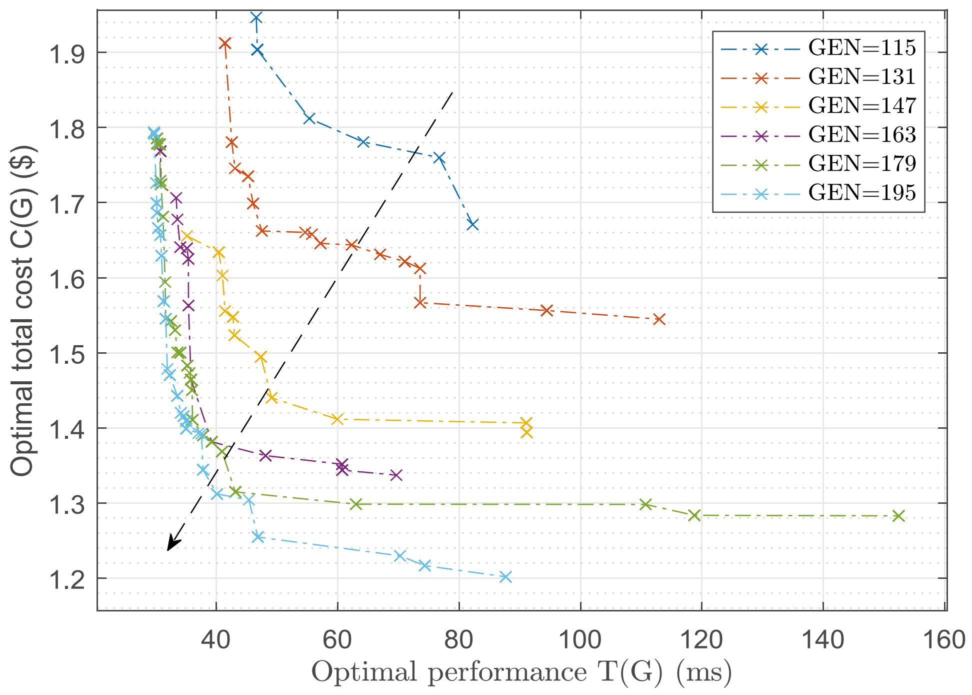

Fig. 5 The convergence with the increasing of iterations

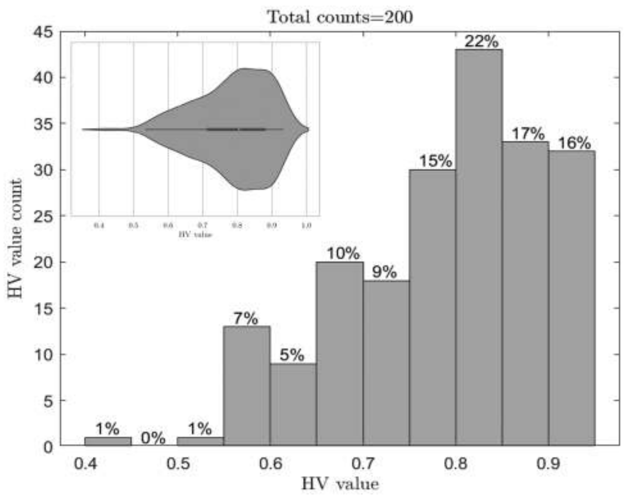

Fig. 7 The distribution of hyper-volume

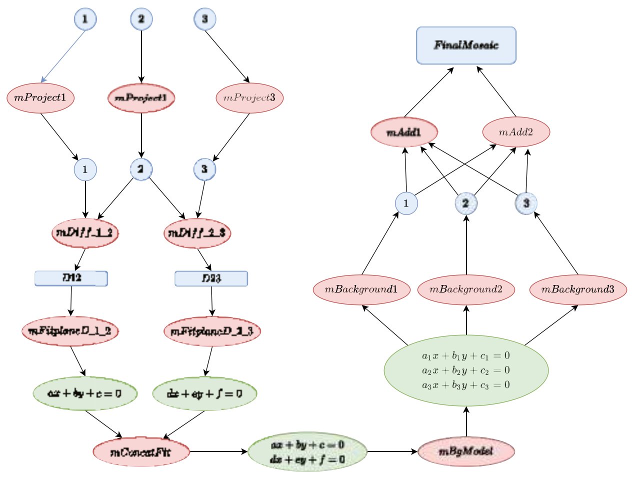

Fig. 8 A directed acyclical graph (DAG) showing the parallelization in the Montage design

Besides the data set from Alibaba Cluster, here we also evaluate our approach on some more complex data structures or workflows like the Montage project[4], which is an astronomical image mosaic engine, to illustrate the good portability of our method. The modules (in red ellipse) of Montage work together in the order shown in Fig. 8. Obviously, it is a more complex service-based application and it can also be deployed in the MEC system. In Fig. 9, we used Montage's DAG data to test our algorithm on the number of services of different scales. With the increasing of service number from montage-1 to montage-3, the Pareto frontiers are shown in Fig. 9. It can be seen from the results that the approach can also work to generate the optimal frontiers in the complex situations.

6.1 Impacts of system configurations

The above comparisons show that the MOEA/D based algorithm will be practical in solving such a context-aware AIoT application deployment problem in MEC-based systems. Besides the comparison between approaches, we will further discuss the effects of various factors in the system.

6.1.1 Impacts of application and service

Among the settings of our MOEA/D based algorithm, the service related factors are the average output data size ($\bar{O}$), the service workload ($\bar{w}$), the average service number (m) and the application type number (K). Therefore, in order to check the influence of these service related parameters on the optimization objective ($T(\boldsymbol{G}), C(\boldsymbol{G})$), we set the system parameters to the configuration listed in Table 2, and adjust the average of the above related parameters respectively to observe their influences. Accordingly, the results are shown in Fig. 10a, b, g, i. That is, the Pareto optimal curves obtained from the algorithm based on MOEA/D shift upward gradually in the direction of performance decrease with the increase of $\bar{O}$, $\bar{w}$, K and m. In detail, with the increase of $\bar{O}$, the Pareto optimal curve gradually drifts toward the direction of performance degradation. This is because when the amount of data increases, it brings more pressure on data transmission between servers. And this pressure will result in traffic congestion and performance deterioration. Secondly, as the average service workload $\bar{w}$ increases, the performance decreases because the increasing workload will force the server to allocate more resources. Thirdly, with the increase of K, which means the increase of applications to be deployed, the Pareto curve moves to the upper right. This is because more service requests need to be processed when the total resources remain the same in this case, and the lack of resources will then reduce the processing efficiency. Finally, similar reasons like those of K decrease the performance with the increasing of service number m for the additional resource requirements.

6.1.2 Impacts of server

Similarly, we keep the system parameters fixed as shown in Table 2, and observe their impacts on the optimization objectives ($T(\boldsymbol{G}), C(\boldsymbol{G})$) by adjusting the server related factors, which are the average deployment price related to the server $\bar{\eta}$, the total number of resources of each server $\bar{\mu}^{\star}$, and the total number of servers n. The results are shown in Fig. 10c, d, h. In summary, the Pareto curve of the optimization objective ($T(\boldsymbol{G}), C(\boldsymbol{G})$) shows a trend to move up and right with the increases of $\bar{\eta}$, and it moves in the direction

[4] http://montage.ipac.caltech.edu/docs/grid.html

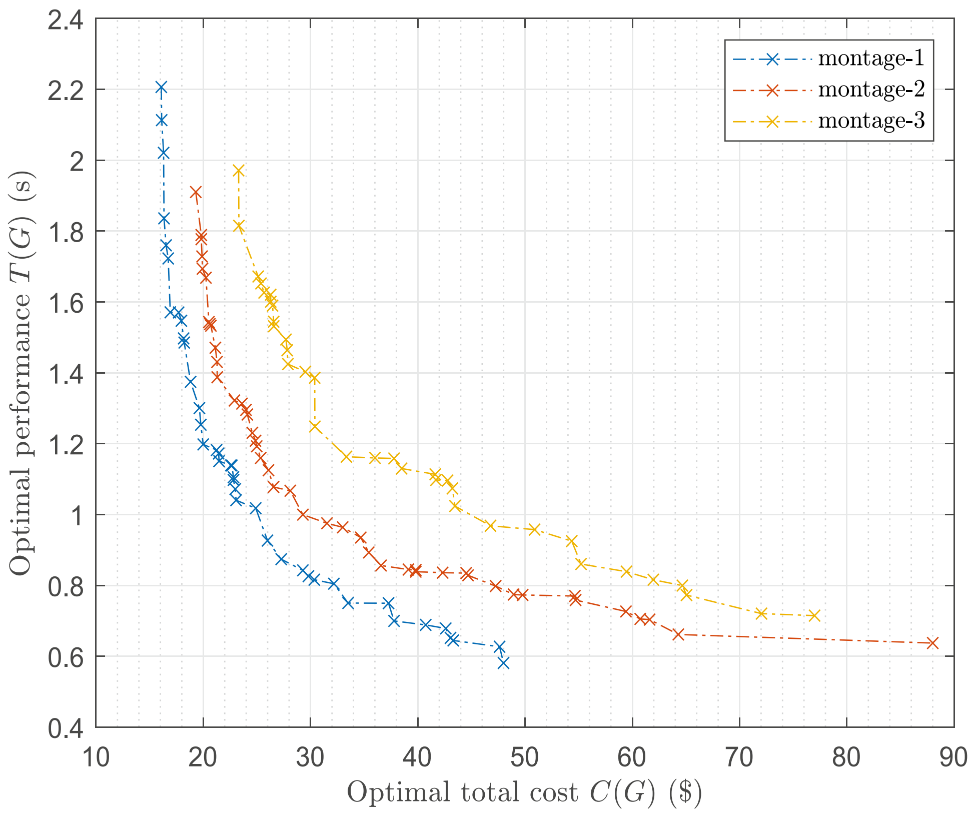

Fig. 9 Result of test on Montage workflow

of enhanced performance as $\bar{\mu}^{\star}$ and n increase. In detail, the Pareto curve moves in the direction of performance decline with the average deployment price increases because the linear relationship between the average price increases and the total cost. Besides, as the average number of resources of the server increases, the performance of the entire system gradually improves. This is due to the relaxation of resource constraints, which can speed up the operation of a single service. However, as the cost increases with more used resources, the curve's trend of moving to the lower left is not particularly obvious. Finally, with the increase in the number of edge servers n, the Pareto curve moves to the lower left. This is because after the increase in the number of servers, while the total number of tasks remains the same, the optional edge servers in the system become diverse, and including some cost-effective servers. Deploying services on these servers will not only shorten the processing time but also lower the cost, so it can improve the overall performance.

6.1.3 Impacts of network

In the same way, we keep the system parameters fixed as shown in Table 2 and adjust the network parameters $\bar{v}^e$ ($\bar{v}^c$), $\bar{b}$ to detect the impacts of network quality on the system. The results are shown in Fig. 10e, f). We can find that the better the network condition is, the closer the Pareto curve of performance and cost is to the lower left. This is because as the network transmission rate increases, the data transmission time cost is reduced with other system parameters unchanged, so that each service can respond faster, which leads to performance improvements and make the Pareto curve move to the lower left.

7 Conclusion and future work

In this paper, we investigate, model and formulate the CA^{3}D problem in resource-constrained MEC environment. A series of numeric experiments are conducted based on the Alibaba and Montage dataset. Since the CA^{3}D problem is a mixed integer nonlinear programming multi-objective problem, which brings enormous challenges to find its optimal solutions. As a compromise scheme, we turn to the heuristic method like MOEA/D and try to find some sub-optimal solutions that may satisfy the requirements at a certain degree within the mentioned constraints. The comparison results show that our algorithm outperforms other representative baseline approaches. And the factor exploration shows how the system settings will become the bottlenecks.

Obviously, this application modeling and problem solutions for CA^{3}D can be transformed into any other application in which the components have partial order dependence as a

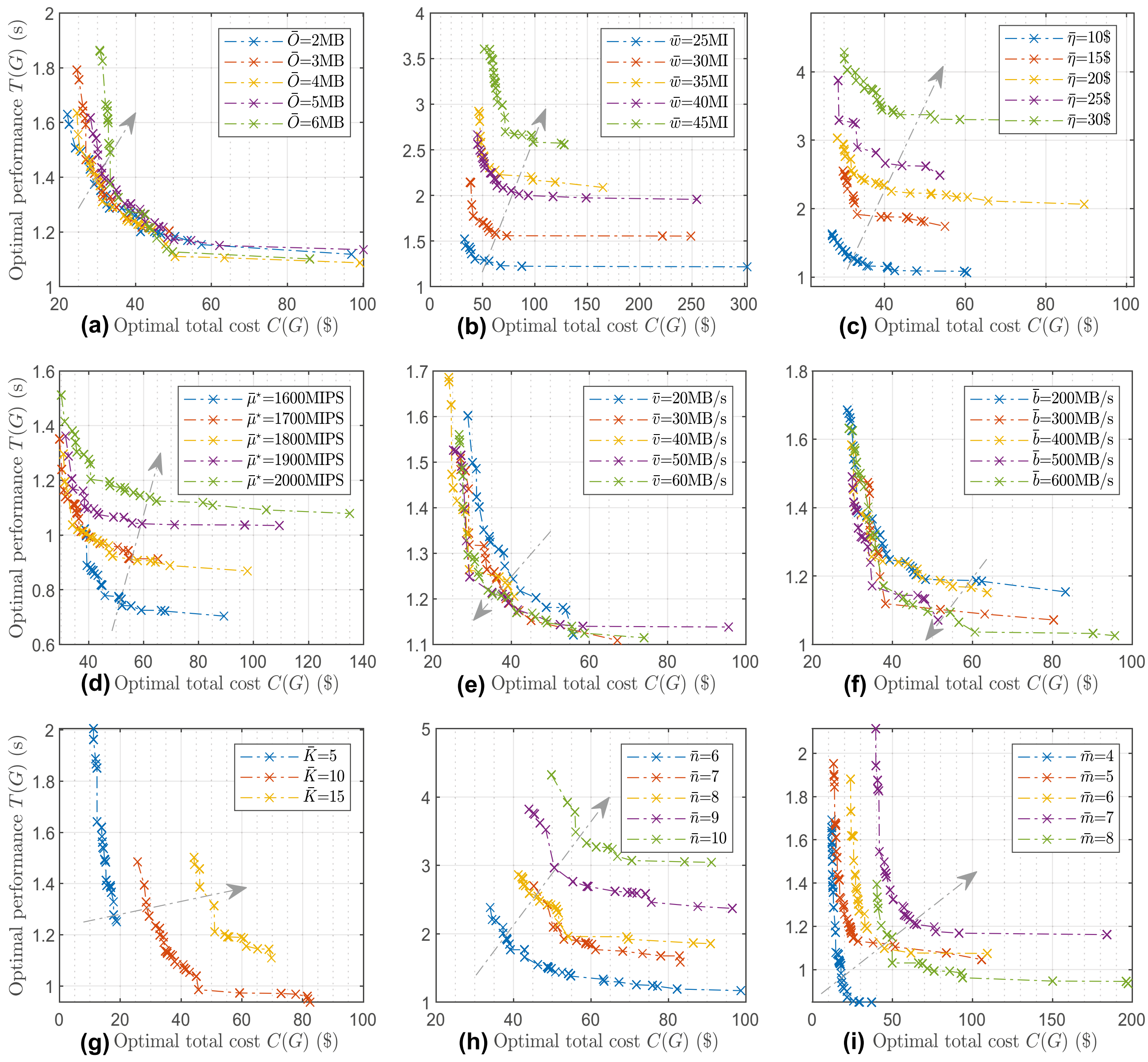

Fig. 10 The impacts of service and edge server settings

directed acyclic graph based on the demands of extensions. It has a instructive significance on how to optimize the deployment of components and resource allocation in such applications so as to achieve the optimal balance between performance and cost — in other words, it has a good compatibility.

However, even the proposed solution can be practical in placing and allocating, it just aims to provide a good start for the system — when the system is established in a very unstable environment, the proposed solution may not guarantee its efficiency (for example, in a context fast-changing environment). Therefore, we are going to consider the uncertainty of real-time scheduling tasks and try to balance the effectiveness and robustness to make the system more self-adaptive in our future work, where the deep reinforcement learning (DRL) based methods may play an important role.

Acknowledgements This research was partially supported by the National Natural Science Foundation of China (No. 62102350, No. 62072402) and the Natural Science Foundation of Zhejiang Province (No. LQ21F020007, No. LQ20F020015).

Declarations

Conflict of interest The authors declare that they do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

References

1. Deng S, Zhao H, Fang W, Yin J, Dustdar S, Zomaya AY (2020) Edge intelligence: The confluence of edge computing and artificial intelligence. IEEE Internet Things J 7(8):7457–7469
2. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: Vision and challenges. IEEE Internet Things J 3(5):637–646
3. Xiang Z, Deng S, Jiang F, Gao H, Tehari J, Yin J (2020) Computing power allocation and traffic scheduling for edge service provisioning. In 2020 IEEE International Conference on Web Services (ICWS), IEEE, pp. 394–403
4. Filippini I, Sciancalepore V, Devoti F, Capone A (2017) Fast cell discovery in mm-wave 5g networks with context information. IEEE Trans Mob Comput 17(7):1538–1552
5. Wang D, Xu D, Yu D, Xu G (2021) Time-aware sequence model for next-item recommendation. Appl Intell 51(2):906–920
6. Fan Q, Ansari N (2018) Application aware workload allocation for edge computing-based iot. IEEE Internet Things J 5(3):2146–2153
7. Chen Y, Deng S, Ma H, Yin J (2020) Deploying data-intensive applications with multiple services components on edge. Mob Netw Appl 25(2):426–441
8. Zhao H, Deng S, Zhang C, Du W, He Q, Yin J (2019) A mobility-aware cross-edge computation offloading framework for partitionable applications. In 2019 IEEE International Conference on Web Services, ICWS, IEEE, pp. 193–200
9. Bozorgchenani A, Mashhadi F, Tarchi D, Monroy SS (2020) Multi-objective computation sharing in energy and delay constrained mobile edge computing environments. IEEE Trans Mob Comput 20(10):2992–3005
10. Chen Y, Zhang Y, Wu Y, Qi L, Chen X, Shen X (2020) Joint task scheduling and energy management for heterogeneous mobile edge computing with hybrid energy supply. IEEE Internet Things J 7(9):8419–8429
11. Jiang C, Fan T, Gao H, Shi W, Liu L, Cerin C, Wan J (2020) Energy aware edge computing: A survey. Comput Commun 151:556–580
12. Mashhadi F, Monroy SAS, Bozorgchenani A, Tarchi D (2020) Optimal auction for delay and energy constrained task offloading in mobile edge computing. Comput Netw 183:107527
13. Ouyang T, Zhou Z, Chen X (2018) Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. IEEE J Sel Areas Commun 36(10):2333–2345
14. Pasteris S, Wang S, Herbster M, He T (2019) Service placement with provable guarantees in heterogeneous edge computing systems. In IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, pp. 514–522
15. Roy P, Sarker S, Razzaque MA, Hassan MM, AlQahtani SA, Aloi G, Fortino G (2020) Ai-enabled mobile multimedia service instance placement scheme in mobile edge computing. Comput Netw 182:107573
16. Yuan B, Guo S, Wang Q (2021) Joint service placement and request routing in mobile edge computing. Ad Hoc Netw 120:102543
17. Ning Z, Dong P, Wang X, Wang S, Hu X, Guo S, Qiu T, Hu B, Kwok RY (2020) Distributed and dynamic service placement in pervasive edge computing networks. IEEE Trans Parallel Distrib Syst 32(6):1277–1292
18. Han P, Liu Y, Guo L (2021) Interference-aware online multi-component service placement in edge cloud networks and its ai application. IEEE Internet Things J
19. Yu Y, Zhang J, Letaief KB (2016) Joint subcarrier and cpu time allocation for mobile edge computing. In 2016 IEEE Global Communications Conference (GLOBECOM), IEEE, pp. 1–6
20. Wang C, Liang C, Yu FR, Chen Q, Tang L (2017) Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. IEEE Trans Wireless Commun 16(8):4924–4938
21. You C, Huang K, Chae H, Kim B-H (2016) Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans Wireless Commun 16(3):1397–1411
22. Guo S, Zhang K, Gong B, He W, Qiu X (2021) A delay-sensitive resource allocation algorithm for container cluster in edge computing environment. Comput Commun 170:144–150
23. Bahreini T, Badri H, Grosu D (2021) Mechanisms for resource allocation and pricing in mobile edge computing systems. IEEE Trans Parallel Distrib Syst 33(3):667–682
24. Yang L, Liu B, Cao J, Sahni Y, Wang Z (2019) Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds. IEEE Trans Serv Comput 14(5):1439–1452
25. Mao Y, Zhang J, Letaief KB (2016) Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J Sel Areas Commun 34(12):3590–3605
26. Rabaey JM, Chandrakasan AP, Nikolić B (2003) Digital integrated circuits: a design perspective, vol 7. Pearson education Upper Saddle River, NJ
27. Burd TD, Brodersen RW (1996) Processor design for portable systems. J VLSI Signal Process Syst Signal Image Video Technol 13(2):203–221
28. Rizvandi NB, Taheri J, Zomaya AY (2011) Some observations on optimal frequency selection in dvfs-based energy consumption minimization. J Parallel Distrib Comput 71(8):1154–1164
29. Zhang Q, Li H (2007) Moea/d: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731
30. Ma X, Yu Y, Li X, Qi Y, Zhu Z (2020) A survey of weight vector adjustment methods for decomposition-based multiobjective evolutionary algorithms. IEEE Trans Evol Comput 24(4):634–649
31. Santiago A, Huacuja HJF, Dorronsoro B, Pecero JE, Santillan CG, Barbosa JJG, Monterrubio JCS (2014) A survey of decomposition methods for multi-objective optimization. In Recent Advances on Hybrid Approaches for Designing Intelligent Systems. Springer, pp. 453–465
32. Katoch S, Chauhan SS, Kumar V (2021) A review on genetic algorithm: past, present, and future. Multimed Tools Appl 80(5):8091–8126

Zhengzhe Xiang received the B.S. and Ph.D. degrees of Computer Science and Technology in Zhejiang University, Hangzhou, China. He was previously a visiting student worked at the Karlstad University, Sweden in 2018. He is currently a lecturer with Zhejiang University City College, Hangzhou, China. His research interests lie in the fields of Service Computing, Cloud Computing, and Edge Computing. He serves as a reviewer for a series of international journals like IEEE Transactions on Services Computing, IEEE Transactions on Mobile Computing, Mobile Networks & Applications, IEEE Intelligent Transportation Systems Transactions, Wireless Networks, IET Communications .etc and he also works as PC members of several international conferences.

Yuhang Zheng is now working toward the M.S. degree in the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include Internet of Things technology, Edge Computing, Service Computing, and Reinforcement Learning.

Mengzhu He received her M.S. degree of Plant Protection in Zhejiang University, Hangzhou, China. She is now a scientific assistant in the Intelligent Plant Factory of Zhejiang Province Engineering Lab in Zhejiang University City College and servers as the lab supervisor in the Hangzhou Industry Brain Laboratory. Her research interests include Plant Virology, Internet of Things technology and Intelligent Agriculture.

Longxiang Shi received his B.Sc in software engineering from Northwestern Polytechnical University, Xi'an, China in 2012. He got his Ph.D. degree in College of Computer Science and Technology, Zhejiang University, Hangzhou, China in 2020. In 2018, he was a Joint Ph.D student with the Advanced Analytics Institute, University of Technology Sydney, Sydney, Australia under the supervision of Professor Longbing Cao. He is currently a lecturer in Zhejiang University City College. His research interests include reinforcement learning, data mining and machine learning.

Dongjing Wang received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 2012 and 2018, respectively. He was co-trained at the University of Technology Sydney, Ultimo, NSW, Australia, for one year. He is currently a Lecturer with Hangzhou Dianzi University, Hangzhou. His current research interests include recommender systems, machine learning, data mining, and business process management.

Shuiguang Deng is currently a full professor at the College of Computer Science and Technology in Zhejiang University, China, where he received a BS and PhD degree both in Computer Science in 2002 and 2007, respectively. He previously worked at the Massachusetts Institute of Technology in 2014 and Stanford University in 2015 as a visiting scholar. His research interests include Edge Computing, Service Computing, Cloud Computing, and Business Process Management. He serves for the journal IEEE Trans. on Services Computing, Knowledge and Information Systems, Computing, and IET Cyber-Physical Systems: Theory & Applications as an Associate Editor. Up to now, he has published more than 100 papers in journals and refereed conferences. In 2018, he was granted the Rising Star Award by IEEE TCSVC. He is a fellow of IET and a senior member of IEEE.

Zengwei Zheng received the B.S. and Master degrees in computer science and western economics from Hangzhou University, China, in 1991 and 1998, respectively, and the Ph.D. degree in computer science and technology from Zhejiang University, China, in 2005. He is currently a Professor with the School of Computer and Computing Science, the Director of Intelligent Plant Factory of Zhejiang Province Engineering Laboratory, the Director of the Hangzhou Key Laboratory for IoT Technology and Application, Zhejiang University City College. His research interests include wireless sensor networks, location-based service, Internet of Things, digital agriculture, and pervasive computing. He is a senior member of CCF.