IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 24, 2020, accepted August 23, 2020, date of publication August 31, 2020, date of current version September 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3020233

Task Allocation Mechanism of Power Internet of Things Based on Cooperative Edge Computing

QIANJUN WANG, SUJIE SHAO, (Member, IEEE), SHAOYONG GUO, XUESONG QIU, (Senior Member, IEEE), AND ZHILI WANG

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding authors: Sujie Shao (buptssj@bupt.edu.cn) and Zhili Wang (zlwang@bupt.edu.cn)

This work was supported in part by the Beijing Natural Science Foundation through the Research on Adaptive Fault Recovery Mechanism for Electric Power IoT under Grant 4194085, and in part by the Fundamental Research Funds for the Central Universities under Grant 2019RC08.

ABSTRACT Edge computing can be widely used in unmanned aerial vehicle (UAV) inspection, field operation control, power consumption information collection and other businesses in the power Internet of Things scene. Edge computing offloads functions such as data processing and applications to network edge nodes near the terminals to provide low-latency services and ensure service quality. However, with the explosive growth of business terminals, the capacity of single edge node is limited and it is difficult to meet all business requirements at the same time. Therefore, this article proposes a task allocation mechanism based on cooperative edge computing. Firstly, a task allocation model based on cooperation of two edge nodes is established to minimize the average task completion delay while meeting business requirements. Secondly, the Two-edge-node Cooperative-task Allocation based on Improved Particle Swarm Optimization (TCA-IPSO) algorithm is proposed, which applies the crossover and mutation strategy in genetic algorithm to improve the particle swarm optimization algorithm, and solves the problem that the task allocation scheme in cooperation is prone to fall into a local optimum. Finally the simulation results show that the proposed TCA-IPSO algorithm reduces the average task completion delay by 53.8% and 36.0% compared to the benchmark and QoS-based Task Distribution (QBTD) algorithm.

INDEX TERMS Cooperative edge computing, power Internet of Things, task completion delay, task allocation.

I. INTRODUCTION

The power Internet of Things is the application of the internet of things (IoT) in the smart grid. It effectively integrates communication infrastructure resources and power system infrastructure resources, realizes the interconnection of all things in the power system, comprehensive state perception and efficient information processing. Power Internet of Things is an example of Energy Internet [1] in power companies. They aim to build a new open and shared energy ecology. But they have different perspectives. The power Internet of Things starts from electricity and then goes into the comprehensive energy beyond electricity. Energy Internet starts from the energy nodes such as power network and natural gas network, and interconnects them to form a shared network.

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

The power Internet of Things provides various services such as video monitoring, sensing detection and equipment inspection [2], [3]. Generally, these business requirements are diversified, such as video monitoring and equipment diagnosis, which require high resources while smart meter monitoring and inspection robots are sensitive to delay and require timely calculation results. With the construction and continuous development of the power Internet of Things, the amount of business terminals' data has shown explosive growth. As a result, the pressure on network transmission and cloud center load has increased under the cloud computing model. Processing delays are also difficult to meet most business requirements.

In order to solve the problem, edge computing has been applied in the power Internet of Things as an extended solution of cloud computing [4]. The power Internet of Things architecture based on edge computing is shown in Figure 1. Deploy edge nodes with computing and storage capabilities

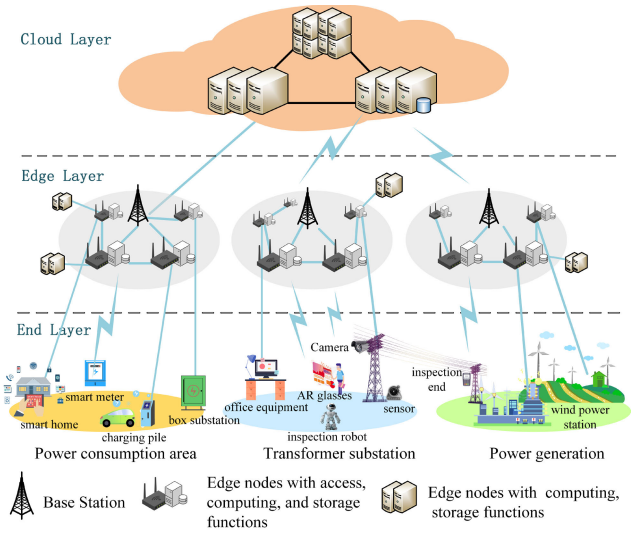


FIGURE 1. The power IoT structure based on edge computing.

such as wireless access points, routers, software defined network (SDN) switches, and edge servers at the edge of the network. Power business terminals are connected to edge nodes through wired, Wi-Fi, micro-power wireless, 4G / 5G, and low-power wide area networks. Putting computing tasks at the edge nodes can reduce network transmission and cloud load.

Compared with the cloud center, the resources of edge nodes are still very limited. As the number of business terminals grows, it is difficult for edge nodes to simultaneously meet the different needs of power IoT services. At the same time, there are differences in load between edge nodes. First, the geographical distribution of business terminals is unbalanced when they connected to edge nodes. Second, in sudden situation a large number of information collection terminals upload data for processing. As a result, some edge nodes need to process lots of business requests while other edge nodes are relatively idle. Therefore, the unbalanced time and space distribution of business requests will further exacerbate the severity of the problem.

In order to solve the problems, the cooperative edge computing can be adopted. The tasks of the single business terminal are offloaded to multiple edge nodes for computing so as to optimize resource usage and reduce the service delay. In video data analysis services, video acquisition terminal uploads a large amount of video data when a substation is in a sudden fire situation. Multiple edge nodes can cooperate to process large amounts of video data. In intelligent inspection services, when multiple inspection robots access to the same edge node, the edge node resources are insufficient to meet requirements of multiple robots at the same time. So other edge nodes are required to cooperate to process the collected data. The work in [5], [6] shows that cooperative computing of edge nodes is better than offloading tasks to a single edge node. References [7], [8] propose cooperative computing of edge nodes, in which the main concern is how to perform optimal task allocation and resource allocation for reducing task completion delay. Reference [9] focuses on network traffic scheduling while considering task scheduling issues, and it models the joint issues to minimize overall completion latency. Reference [10] integrates the horizontal offloading of edge nodes in the cloud-edge-end three tier computing offloading framework to improve resource utilization. Reference [11] considers the dependencies between tasks in task assignment and proposes a complete polynomial time approximation scheme to solve the optimal task assignment. In these literatures, horizontal cooperation balances computing load and reduces business service delays. However, the above methods do not consider the multiple requirements of business at the same time, including computing resource requirement, storage resource requirement and delay requirement. Some literatures only consider how to compute cooperatively for a single request instead of multiple requests.

Aiming at the problem of limited edge node resources and unbalanced load that make business requirements difficult to meet in the power IoT scenario, we propose a task allocation mechanism based on cooperative edge computing to minimize the average task completion delay under the constraints of business requirements. More specifically, the main contributions are as follows:

- In order to make better use of edge node resources and reduce business completion delay, a task allocation model based on cooperation of two edge nodes is proposed. In the model, the business request and resource model and delay model are established respectively, and then cooperative computing model is built, including three types: independent computing of access points, cooperative computing of access point and neighbor edge nodes, and cooperative computing of neighbor edge nodes, which aim to shorten the average task completion delay under the constraints of business requirements.
- The model is transformed into an integer non-linear programming problem, and the TCA-IPSO algorithm is proposed for solving. The algorithm uses the crossover and mutation operations of the genetic algorithm to improve the particle update strategy in the particle swarm optimization algorithm. While the particles maintain learning ability, the diversity of the particle population is improved and precocity is prevented from falling into a local optimum.

The rest of the paper is organized as follows: The chapter II reviews the related work of cooperative edge computing. The chapter III establishes the task allocation model based on the cooperative edge computing. The chapter IV introduces the TCA-IPSO algorithm. The chapter V shows the performance of proposed algorithm by simulation. The chapter VI summarizes the paper and draws conclusions.

II. RELATED WORK

Cooperative edge computing mainly studies how business requests are allocated and completed in the edge network

to provide satisfactory services. Aiming at the problems of insufficient resources at the edge of the network, uneven time and space distribution of business requests, unreasonable task allocation and insufficient life span of edge nodes, there are four types of edge cooperation study: 1) Edge cooperation with latency as the optimization goal. It is mainly targeted at service scenarios with high delay requirements. 2) Edge cooperation technology with energy consumption as the optimization goal. It mainly considers that some edge devices have limited battery capacity due to their portability. It focuses on the energy consumption problem and aims to minimize the overall energy in the case of business response time constraints. 3) Edge cooperation technology that starts from service response time and energy consumption. It realizes joint optimization of factors such as delay and energy consumption. 4) Edge computing with network performance as optimization goal including network throughput, load balancing of edge nodes.

Latency is a key point in the research of cooperative edge computing. Reference [10] proposes a cooperative offloading framework in a three-tier mobile edge computing network. At the same time, it focuses on horizontal offloading between edge nodes. Through the joint optimization of offloading decisions and computing resource allocation, it minimizes the average task duration in the case of limited equipment power. Reference [11] studies the task assignment problem with task dependency, minimizes service delay under the constraint of resource cost, and finally provides an approximate polynomial algorithm to solve the problem. Aiming at the problem of computing offload in fog computing networks, [12] proposes a fog node (FN) cooperation strategy for offloading problem, which translates the offloading problem into a workload allocation problem for minimizing services delay at a given power efficiency. Then a parallel optimization framework based on alternating direction method of multipliers is used to solve the problem and improve the network performance of fog computing. Reference [13] proposes a task allocation mechanism based on edge computing, which assigns different types of tasks to the corresponding virtual machines (VM) in each cloudlet, and finds the optimal VM resource allocation strategy to minimize the average response delay of the application.

Although allocating tasks on edge devices for processing can guarantee the real-time advantages of edge computing environments, generally mobile edge devices have limited battery capacity. For this reason, the energy consumption of devices need to be optimized. Reference [7] considers a three-node mobile edge computing (MEC) system including user nodes, auxiliary nodes, and access point (AP) nodes.Then it develops an energy-saving design framework for partial and binary offloading. Reference [14] studies the cooperation between mobile devices and MEC in the use scenario of wearable devices, and builds a two-layer task unloading model to minimize the energy consumption of Mobile devices and MEC under the constraint of service's delay. Reference [15] studies cooperative communication method in the wireless transmission MEC system of two users to minimize AP transmission energy, and a two-stage method is proposed to obtain the optimal resource allocation strategy.

Cooperative edge computing should not only consider the time delay of task completion, but also consider the energy consumption factor of edge node as the same time. Reference [16] establishes a cooperative computing system to handle user offloading work. They share fog network resources through FN cooperative forwarding. Then a joint energy and time cost minimization problem is proposed. Finally it designs a low complexity of fairness cooperation algorithm (FCA) to solve the optimization problem. Reference [17] proposes Mobile Edge Computing-Base Station (MEC-BS) cooperation strategy, which offloads queued tasks on MEC to other MEC-BS directly connected to enhance service satisfaction, and finally converts it into the problem of maximizing the total time and energy consumption. Reference [18] proposes a lightweight computing offloading method, which can improve the performance of wearable devices and reduce energy consumption by distributing computing tasks of wearable devices to multiple nearby mobile devices.

There are also some literatures that take network throughput and load balancing as optimization goals. Reference [8] proposes a communication-aware chained task scheduling method that takes into account the large amount of communication costs in dispersed computing. It proposes a virtual queuing network that encodes the state of the network, and uses the Max-Weight type scheduling strategy to achieve optimal network throughput. Finally, the scheduling problem is extended to the directed acyclic graph (DAG) task model. Reference [19] proposes a two data center cooperation scheme for fog or edge computing environments. When the data center buffer is full, the upcoming tasks are migrated to adjacent data center. At the same time, each data center adopts the same strategy to handle the task, which minimizes the blocking state of each edge node.

In addition, there are some literatures discussing collaborative edge computing from some novel perspectives recently. Reference [20] proposes an edge computing framework to enable cooperative processing on resource-abundant mobile devices for delay-sensitive multimedia IoT tasks. It optimally forms mobile devices into video processing groups and dispatch video chunks to proper video processing groups. Reference [21] studies the multi-hop computation-offloading problem for the Industrial Internet of things (IIoT)-edge-cloud computing model and adopts a game-theoretic approach to achieving quality of service (QoS)-aware computation offloading in a distributed manner. The proposed algorithm offers a stable performance gain for IIoT in various scenarios. Reference [22] considers the problem of cooperative computation offloading for UAVs, which optimizes the transmission data rate and resource allocation to satisfy different QoS requirements. The proposed algorithm can efficiently utilize heterogeneous edge servers in a cooperative manner. Reference [23] proposes a novel Intelligent

Cooperative Edge (ICE) computing framework, which realizes the complementary integration of edge computing and artificial intelligence (AI) in the IoT environment from two aspects: edge-based AI and AI-enabled edge. Reference [24] studies the joint optimization of the CPU frequencies, the offloading bits, the transmit power, and the UAV's trajectory of the UAV-enabled wireless powered cooperative MEC system. An optimization problem is formulated to minimize the required energy of UAV. A successive convex approximation (SCA)-based algorithm-based algorithm and a decomposition and iteration (DAI)-based algorithm are proposed to tackle the nonconvex problem. Reference [25] proposes a novel paradigm of socially-motivated cooperative mobile edge computing, which leverages the social tie structure among mobile and wearable device users for achieving effective and trustworthy cooperation in task executions.

Particle Swarm Optimization (PSO) is a random search algorithm based on group cooperation, which is developed by simulating foraging behavior of birds. The algorithm is often used in the optimization problem recently. Reference [26] proposes nonlinear exponential inertia weight PSO algorithm which is used to get solution in the edge-cloud collaborative multi-task computing unloading model. By dynamically adjusting the inertia weight, the algorithm can make up for the convergence premature defect of the standard PSO, and effectively avoid falling into the local optimal solution. Reference [27] proposes a task offloading scheme merely relying on vehicle-to-vehicle communication, which is further solved by the PSO algorithm. The standard particle position update formula is used, and the problem of particles not meeting the constraints is solved through adjustment. Reference [28] proposes ILCDPSO algorithm which improves the convergence speed of particle swarm algorithm by adding local search strategy and chaotic sequence. Genetic algorithm (GA) is also widely used in solving optimization problems. Reference [29] proposes an improved GA in which the crossover probability and mutation probability are dynamically adjusted according to the change of individual fitness, and the convergence speed of the algorithm is accelerated. A number of hybrid algorithms of GA and PSO also have been proposed. Reference [30] designs a suboptimal algorithm named as hierarchical GA and PSO-based computation algorithm, in which GA and PSO are executed alternately until the number of iterations is satisfied. Reference [31] proposes HGAPSO algorithm based on the idea of updating particle positions in the particle swarm. The next generation chromosome is modified by recording the historical optimality of each chromosome and population optimality, which is applied to change the mutation operation rule.

III. TASK ALLOCATION MODEL BASED ON COOPERATION OF TWO EDGE NODES

The flowchart of the task allocation mechanism based on the cooperative edge computing is shown in Figure 2. First, a task allocation model based on cooperation of two edge nodes is established based on all business requests and the idle resources of the edge nodes at a certain moment. We analyze business requests, resources, delay and cooperative computing separately. Then we build task allocation problem to minimize the average task completion delay. Finally we propose the TCA-IPSO algorithm to solve the problem. The TCA-IPSO algorithm improves the standard particle swarm algorithm and introduces the crossover and mutation operations into the particle update strategy so that the particles approach the optimal solution.

TABLE 1. Definition of variables used in the article.

Symbol	Description
N	the number of UEs
M	the number of ENs
$\mathcal{N}$	UE set
$\mathcal{M}$	EN set
R_i	EN set for completing subtasks of UE i
$\mathcal{T}_i$	subtask set requested by UE i
w_{ij}	subtask j of UE i
c_{ij}	the computing resource requirement of w_{ij}
e_{ij}	the storage resource requirement of w_{ij}
d_{ij}	the amount of input data of w_{ij}
t_{ij}	the computing delay of w_{ij} if the resource requirement is met
$\hat{t}_i$	the delay constraint of $\mathcal{T}_i$
λ_{ij}	the ratio of the result data amount to the input data amount
C_k	the number of virtual computing units of EN k
E_k	the number of virtual storage unit of EN k
x_{ijk}	whether subtask j of UE i is allocated to EN k
u_i	the access point of UE i
$\mathcal{N}_k$	the UE set associated with EN k
B_k	the bandwidth of EN k
p_i	the transmission power of UE i
h_{i,u_i}	the UE i and ENu_i channel gain
σ^2	the additive Gaussian white noise power
y_i	the signal-to-noise ratio of UE i and EN u_i
$v_{k,k'}$	the data transmission rate from EN k to EN k'
$T^{finish}_{i,r_{i1},r_{i2}}$	the cooperative completion delay of subtask $\mathcal{T}_i$ by r_{i1}, r_{i2}
$T^{comm}_{i,r_{i1},r_{i2}}$	the data transmission delay of subtasks of UE i from EN r_{i1} to EN r_{i2}
$T^{comp}_{i,r_{i1}}$	the computing delay of the subtasks of UE i on EN r_{i1}

A. BUSINESS REQUEST AND RESOURCE

Assume that the number of user end (UE) and edge node (EN) in the network are N, M respectively. EN is an edge device with computing and storage capabilities. $\mathcal{N} = \{1, 2, \ldots, N\}$ and $\mathcal{M} = \{1, 2, \ldots, M\}$ denote UE set and EN set respectively.

The task request of a UE is cooperatively completed by EN collection. Considering that multiple EN cooperation will bring certain communication overhead, we adopt two ENs cooperation method. The decision of the cooperative ENs is represented by R. All the subtask of UE i is completed by EN set $R_i = \{r_{i1}, r_{i2} \in \mathcal{M}\}$.

The subtask set requested by UE i is $\mathcal{T}_i$, where the subtask j is represented by $w_{ij} = (c_{ij}, e_{ij}, d_{ij}, t_{ij}, \lambda_{ij})$. c_{ij} represents the computing resource requirement, e_{ij} represents the storage resource requirement, d_{ij} represents the amount of input data, t_{ij} represents the computing delay if the resource requirement is met, and λ_{ij} represents the ratio of the result data amount to the input data amount. There is no temporal dependency

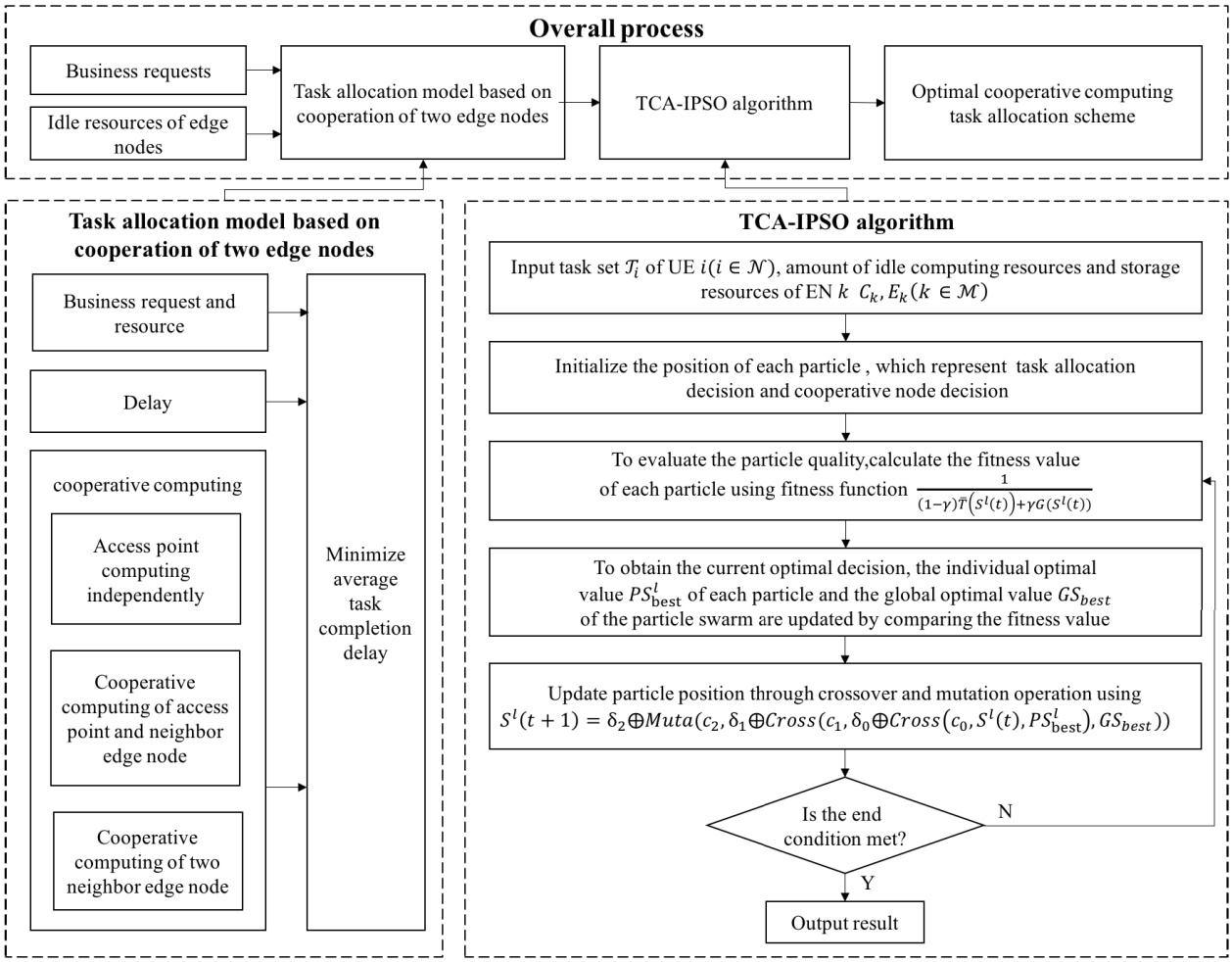


FIGURE 2. Flowchart of the whole method.

between subtasks, so they can be completed independently. $\hat{t}_i$ represents the delay constraint of $\mathcal{T}_i$. Considering the resources of edge nodes are heterogeneous, we use container and virtualization technologies to support the implementation of resource allocation. The amount of required resources is represented by the number of virtual resource units.

Assume that all UE requests are sent simultaneously at a certain time. (C_k, E_k) represents the idle resources of EN k. C_k and E_k indicate the number of idle virtual computing units and idle virtual storage units respectively. The task allocation decision $X = \{x_{ijk}\}$ is specified as follow,

$$x_{ijk} = \begin{cases} 1 & \text{if subtask } j \text{ of UE } i \text{ is allocated to EN } k \\ 0 & \text{else} \end{cases} \tag{1}$$

A subtask of UE i can only be executed by one EN in R_i, so it has the following constraints

$$\sum_{k \in \mathcal{M}} x_{ijk} = 1 \tag{2}$$

$$j \in R_i, \ \forall x_{ijk} = 1 \tag{3}$$

EN needs to meet the computing and storage resource requirements of the subtasks that allocated to itself. $Sum^{comp}_{R,X,k}$ and $Sum^{sto}_{R,X,k}$ represents the amount of EN k computing and storage resource units that should be satisfied under cooperative node decision R and task allocation decision X. Therefore, the constraints is as follow,

$$Sum^{comp}_{R,X,k} = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{T}_i} c_{ij} x_{ijk} \le C_k, \quad k \in \mathcal{M} \tag{4}$$

$$Sum^{sto}_{R,X,k} = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{T}_i} e_{ij} x_{ijk} \le E_k, \quad k \in \mathcal{M} \tag{5}$$

B. DELAY

Each UE accesses the nearest EN. We assume that ENs and UEs will not move. EN $u_i \in \mathcal{M}$ represents the access point of UE i, so the UE set associated with EN k can be represented as $\mathcal{N}_k = \{i : i \in \mathcal{N}, u_i = k\}$.

The bandwidth resource of EN k is B_k Hz. We default that EN bandwidth is allocate by the UEs associated with the EN evenly. There is no other interference between UEs when they connect to the same EN. We denote the uplink spectral efficiency between UE i and EN u_i as η_i. It can be approximated by Shannon's formula $\eta_i = log(1 + y_i)$. The uplink signal-to-noise ratio of UE i is $y_i = \frac{p_i h_{i,u_i}}{\sigma^2}$, where

p_i is the transmission power of UE i, h_{i,u_i} is channel gain between UE i and ENu_i, and σ^2 is additive white Gaussian noise power. Therefore when the UE i accesses the EN u_i, the uplink transmission rate of the UE i for radio access can be given by

$$v_i = \frac{B_{u_i}}{|\mathcal{N}_{u_i}|} log(1 + y_i) \tag{6}$$

where $|\mathcal{N}_{u_i}|$ denotes the number of UEs that connected to the EN u_i.

Similar to [32], [33], the downlink bandwidth of the UE i is much higher than the uplink bandwidth, and data size after task processing is usually much smaller than it before processing, so we ignore the downlink transmission delay of sending the task results from ENs u_i to UE i.

We denote the data transmission rate from EN k to EN k' as $v_{k,k'}$. Similar to [34], [35], it is assumed that the data transmission rate can be obtained by measurement.

C. COOPERATIVE COMPUTING

The subtasks of UE i are completed cooperatively by EN set $R_i = \{r_{i1}, r_{i2} \in \mathcal{M}\}$. r_{i1} and r_{i2} process part of them. The subtask results are aggregated at single EN in R_i and finally returned to the access point EN u_i. According to whether EN u_i participates in cooperation, there are three types of cooperation methods: 1). Access point computing independently, $r_{i1} = r_{i2} = u_i$.2). Cooperative computing of access point and neighbor EN, $r_{i1} = u_i, r_{i2} \neq u_i$ 3). Cooperative computing of two neighbor ENs, $r_{i1} \neq u_i, r_{i2} \neq u_i, r_{i1} \neq r_{i2}$. The three cooperation methods are shown in Figure 3. In the two-edge-nodes cooperation, the cooperative nodes are the nodes which participate in the task computing, excluding the nodes for task and processed result forwarding.

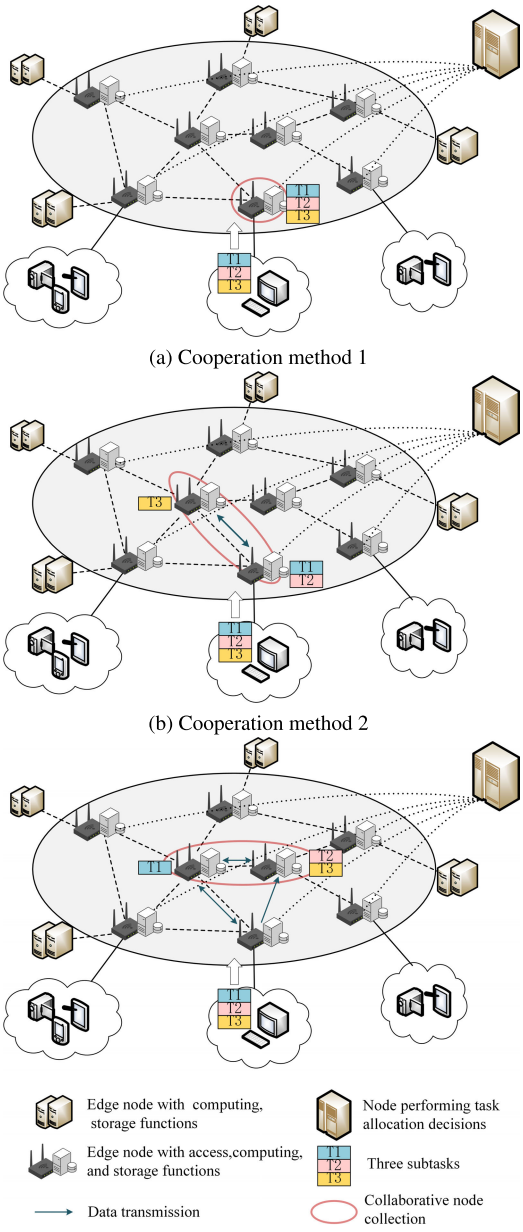
Figure 3(a) shows the cooperation method 1, in which access point completes the subtasks T1, T2 and T3 sent by the UE. Figure 3(b) shows the cooperation method 2, in which the access point completes the subtask T1 and T2, and another neighbor EN completes the subtask T3, and the processed results finally are merged in the access point. Figure 3(c) shows the cooperation method 3, in which two neighbor ENs complete, and one merges the processed results and returns them to access point.

$T^{finish}_{i,r_{i1},r_{i2}}$ represents the cooperative completion delay of UE i, which includes the communication delay of task input data sent from EN u_i to EN r_{i1} and r_{i2}, the computing delay on EN r_{i1} and EN r_{i2}, and the delay of results merging and returning to EN u_i.

1) ACCESS POINT COMPUTING INDEPENDENTLY

The subtasks of the UE i are all completed by the EN u_i, so the cooperative completion delay of the UE i is

$$T^{finish}_{i,r_{i1},r_{i2}} = T^{comp}_{i,u_i} = \sum_{j \in \mathcal{T}_i} t_{ij} = 1 \tag{7}$$



(a) Cooperation method 1

(b) Cooperation method 2

(c) Cooperation method 3

FIGURE 3. Three cooperation methods.

2) COOPERATIVE COMPUTING OF ACCESS POINT EN u_i AND NEIGHBOR EN r_{i2}

First, the EN u_i completes some subtasks of the UE i, and another part are sent to the neighbor EN r_{i2} for computing. After the subtask computing on the EN r_{i2} is completed, the results are returned to the EN u_i for merging. The computing delays of the subtasks on EN u_i and EN r_{i2} are $T^{comp}_{i,u_i} = \sum_{j \in \mathcal{T}_i} t_{ij} x_{iju_i}$, $T^{comp}_{i,r_{i2}} = \sum_{j \in \mathcal{T}_i} t_{ij} x_{ijr_{i2}}$.

The data transmission delay from EN u_i to EN r_{i2} is $T^{comm}_{i,u_i,r_{i2}} = \sum_{j \in \mathcal{T}_i} \frac{d_{ij} x_{ijr_{i2}}}{v_{u_i,r_{i2}}}$, and the transmission delay of the

computing result back to u_i is $T^{comm}_{i,r_{i2},u_i} = \sum_{j\in\mathcal{T}_i} \frac{\lambda_{ij} d_{ij} x_{ijr_{i2}}}{v_{r_{i2},u_i}}$. So the cooperative completion delay of the UE i is

$$T^{finish}_{i,r_{i1},r_{i2}} = max\{T^{comp}_{i,u_i}, T^{comm}_{i,u_i,r_{i2}} + T^{comp}_{i,r_{i2}} + T^{comm}_{i,r_{i2},u_i}\} \quad (8)$$

3) COOPERATIVE COMPUTING OF TWO NEIGHBOR EN

EN u_i divides the subtasks into two parts and forwards them to EN r_{i1} and r_{i2} respectively. Finally, the processed results are merged on r_{i1} or r_{i2} and sent back to EN u_i. The results are merged on the last EN that completes its subtask. We denote the EN as r_{i1}.

The communication delays of EN u_i sending the subtasks of to EN r_{i1} and r_{i2} are $T^{comm}_{i,u_1,r_{i1}} = \sum_{j\in\mathcal{T}_i} \frac{d_{ij} x_{ijr_{i1}}}{v_{u_i,r_{i1}}}$ and $T^{comm}_{i,u_i,r_{i2}} = \sum_{j\in\mathcal{T}_i} \frac{d_{ij} x_{ijr_{i2}}}{v_{u_i,r_{i2}}}$ respectively.

The computation delays on EN r_{i1} and r_{i2} are $T^{comp}_{i,r_{i1}} = \sum_{j\in\mathcal{T}_i} t_{ij} x_{ijr_{i1}}$, $T^{comp}_{i,r_{i2}} = \sum_{j\in\mathcal{T}_i} t_{ij} x_{ijr_{i2}}$ respectively.

The communication delay for sending subtask result from the EN r_{i2} to the EN r_{i1} is $T^{comm}_{i,r_{i2},r_{i1}} = \sum_{j\in\mathcal{T}_i} \frac{\lambda_{ij} d_{ij} x_{ijr_{i2}}}{v_{r_{i2},r_{i1}}}$, and the communication delay for all the subtask results from the EN r_{i1} to the EN u_i after merging is $T^{comm}_{i,r_{i2},u_i} = \sum_{j\in\mathcal{T}_i} \frac{\lambda_{ij} d_{ij}}{v_{r_{i1},u_i}}$.

The subtask results merging time depends on the maximum value of time when the subtask is completed on EN r_{i1} and the time when the subtask result computed by EN r_{i2} has been sent to EN r_{i1}, so the cooperative completion delay for UE is

$$T^{finish}_{i,r_{i1},r_{i2}} = max\{T^{comm}_{i,u_i,r_{i1}} + T^{comp}_{i,r_{i1}}, T^{comm}_{i,u_i,r_{i2}} + T^{comp}_{i,r_{i2}} + T^{comm}_{i,r_{i2},r_{i1}}\} + T^{comm}_{i,r_{i1},u_i} \quad (9)$$

D. PROBLEM MODEL

The total delay from the time when the UE i sends the task request to the time when it receives the calculation result is

$$T_i = t^{up}_{i,u_i} + T^{finish}_{i,r_{i1},r_{i2}} + T^{down}_{i,u_i} = T^{up}_{i,u_i} + T^{finish}_{i,r_{i1},r_{i2}} \quad (10)$$

where $T^{up}_{i,u_i} = \sum_{j\in\mathcal{T}_i} \frac{d_{ij}}{v_i}$ represents the delay of uploading imput data and T^{down}_{i,u_i} represents the delay of returning the processing result from EN u_i to the UE i. As described above, the delay is small and ignored, and the computing delay of execution decision and the transmission delay of returning decision data are also ignored. Therefore, the average completion delay of tasks for all UEs is

$$\overline{T} = \frac{1}{N} \sum_{i\in\mathcal{N}} T_i \quad (11)$$

This article describes the task allocation problem as minimizing the average completion delay of tasks. We hope that the model can consider all users and realize the global optimization of the system. The task allocation decision of the UE is represented by (R, X), $R = \{R_i, i \in \mathcal{N}\}$. Therefore, the task allocation problem in this article is described as follows,

$$\begin{aligned}
P1:\ & Minimize \overline{T} \\
& s.t. \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}_i} c_{ij} x_{ijk} \le C_k, \quad k \in \mathcal{M} && C1 \\
& \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}_i} e_{ij} x_{ijk} \le E_k, \quad k \in \mathcal{M} && C2 \\
& T_i \le \hat{t}_i, \quad i \in \mathcal{N} && C3 \\
& R_i = \{r_{i1}, r_{i2} \in \mathcal{M}\}, \quad i \in \mathcal{N} && C4 \\
& k \in R_i, \quad \forall x_{ijk} = 1,\ i \in \mathcal{N},\ j \in \mathcal{T}_i, \\
& k \in \mathcal{M} && C5 \\
& \sum_{k\in\mathcal{M}} x_{ijk} = 1, \quad i \in \mathcal{N},\ j \in \mathcal{T}_i && C6 \\
& x_{ijk} \in \{0, 1\}, \quad i \in \mathcal{N},\ j \in \mathcal{T}_i,\ k \in \mathcal{M} && C7 \quad (12)
\end{aligned}$$

C1 and C2 indicate that EN needs to meet the subtask's computing and storage resource requirements in task allocation. C3 indicates that each task is to be completed within the time constraint. C4 and C5 indicate that each subtask of the UE i can only be completed by the EN in R_i. C6 means that each task can only be completed by one EN. C7 indicates whether the subtask w_{ij} is completed on EN k, $x_{ijk} = 1$ means yes, $x_{ijk} = 0$ means no.

It is a challenge to ensure that the task allocation scheme meets the resource and delay constraints. That is to say, for each EN it is necessary to ensure that EN can all meet the resource requirements of the tasks assigned to it, and the task allocation decision should meets the task delay requirements. This is an important premise for us to minimize the average delay.

IV. TCA-IPSO ALGORITHM

Standard PSO algorithm is difficult to solve the discrete optimization problem. The initial position and speed of particle are continuous variables, and the particle's position update formula in the algorithm is only suitable for continuous variables. However, the problem proposed in this article is in discrete space. We propose TCA-IPSO algorithm to solve P1, in which crossover and mutation operations in the GA are used to improve the update formula in PSO. The crossover and mutation operations can handle discrete variables well. Reference [28] does not fuse GA and PSO, and the algorithm still retains GA and PSO's own problems such as slow convergence and difficulty in achieving global optimum. The TCA-IPSO embeds the crossover and mutation operations of the genetic algorithm into the PSO algorithm. While preserving the historical memory of particle, the crossover and mutation operations improve the diversity of the population and can better converge to the global optimum.

TCA-IPSO encodes the decision variables R, X in (12), and each particle represents a task allocation scheme. We establish particle update strategy and particle Fitness function based on the input of the problem in (12) (resources of EN, requirements of UE, etc.), which are used to make the parti-

cles evolve towards the optimal direction. Firstly, we encode the problem P1. Secondly we define the Fitness function for evaluating particle quality. Thirdly we describe the particle update strategy, and finally we give the TCA-IPSO algorithm flow.

A. PROBLEM ENCODING

Assume the particle population size is Y. The lth particle represents D-dimensional position vector, denoted as $S^l = (s_1^l, s_2^l, \ldots, s_D^l)$. The optimal position of the lth particle so far is the individual optimal value, denoted as $PS_{best}^l = (p_1^l, p_2^l, \ldots, p_D^l)$, and the optimal position of the whole particle swarm so far is the global optimal value $GS_{best} = (g_1, g_2, \ldots, g_D)$.

We adopt the discrete coding strategy to generate candidate particles, and each particle represent a cooperation decision and task allocation decision. If $X = \{x_{ijk}\}$ is directly used as the position vector of particles in the TCA-IPSO, the vector dimension will be relatively high, which will affect the efficiency of the algorithm. We recode the position vector of particles. The position vector of lth particle after t iteration is expressed as

$$\begin{aligned} S^l(t) &= (R^l(t), Z^l(t)) \\ &= (r_{11}^l(t), r_{12}^l(t), \ldots, r_{N1}^l(t), r_{N2}^l(t), z_{11}^l(t), z_{12}^l(t), \ldots, \\ &\quad z_{N|\mathcal{T}_N|}^l(t)) \end{aligned} \tag{13}$$

where $r_{i1}^l(t), r_{i2}^l(t) \in [1, M]$ means the EN $r_{i1}^l(t)$ and $r_{i2}^l(t)$ cooperate to complete the task $\mathcal{T}_i$ of UE i. So the constraint C4 is met. $z_{ij}^l(t) \in \{1, 2\}$ represents the subtask j of the UE i executed on $r_{iz_{ij}^l}^l(t)$ so the contraints C5, C6 and C7 are met. So in fact $S^l(t)$ is equivalent to the decision vector $X = \{x_{ijk}\}$ of a task allocation scheme. For the sake of convenience, we unify the symbols,

$$S^l(t) = (s_1^l(t), s_2^l(t), \ldots, s_D^l(t)) \tag{14}$$

where $D = 2N + \sum_{i=1}^{N} |\mathcal{T}_i|$.

B. FITNESS FUNCTION

The Fitness function is used to evaluate particle quality and is also an optimization goal in TCA-IPSO algorithm. Since there are constraints in the problem P1, but the evolutionary algorithm is an unconstrained search technology. It cannot guarantee that every particle is always in the feasible region. It is necessary to combine constraint processing technology in the process for solving constrained optimization problem. We construct a constraint violation function through constraint C1-C3 and integrate it into the Fitness function. The constraint violation degree function is as follows,

$$\begin{aligned} G(S) = \sum_{k \in \mathcal{M}} \{max\{Sum_{S,k}^{comp} - C_k, 0\} + max\{Sum_{S,k}^{sto} \\ - E_k, 0\}\} + \sum_{i \in \mathcal{N}} max\{T_i - \hat{t}_i, 0\} \end{aligned} \tag{15}$$

$G(S)$ represents the sum of constraint violations of all EN's computing resources, storage resources and delay. When the particle is within the feasible region, $G(S) = 0$, that is, all the particles satisfying $G(S) = 0$ constitute the feasible region of the search space. When particle is not in the feasible region, $G(S) > 0$. In order to ensure that the problem in (12) can be solved and the constraint C1-C3 are met at the same time, we merge the constraint C1-C3 into the Fitness function. The Fitness function is defined as follow,

$$Fitness(S) = \frac{1}{(1-\gamma)\overline{T}(S) + \gamma G(S)} \tag{16}$$

Larger $G(S)$ or $\overline{T}(S)$ value will result in smaller fitness value which will influence the flight direction of particle. We usually set γ close to 1, in order to ensure that when the constraints C1, C2, and C3 are not satisfied the fitness value will tend towards 0.

C. UPDATE STRATEGY

In the update strategy particles obtain information from historically optimal particles and globally optimal particles. It makes the particles evolve towards the optimal direction.The globally optimal particle obtained is the optimal task allocation scheme. Since the particle uses the discrete encoding method, the update strategy in the standard particle swarm optimization algorithm is no longer applicable. In this article, the crossover and mutation operations in genetic algorithm are used to update the particle position.

$$\begin{aligned} S^l(t+1) = \delta_2 \oplus Mutate(c_2, \delta_1 \oplus Cross(c_1, \delta_0 \oplus Cross(c_0, \\ S^l(t), PS_{best}^l), GS_{best})) \end{aligned} \tag{17}$$

where $Cross()$ and $Mutate()$ represent crossover operation and mutation operation respectively. The specific crossover and mutation processes are shown in Figure 4. In Figure 4 (a), $\hat{S}$ represents the individual optimal particle PS_{best}^l or the global

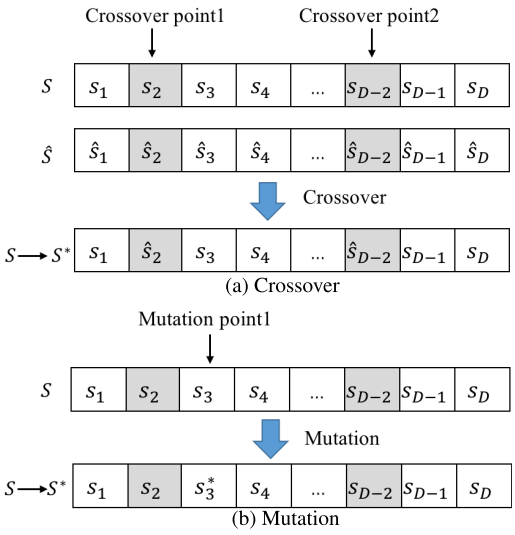


FIGURE 4. Crossover and mutation operation.

optimal particle GS_{best}. It selects some crossover points to cross particle S and $\hat{S}$. In Figure 4(b), some mutation points are selected from S for mutation.

Through crossover operation, each particle can obtain the position information from the historical optimal particle and the global optimal particle directly, thereby converging to the historical optimal and global optimal. The mutation operation can improve the diversity of particles and prevent them from falling into local optimum.

1) CROSSOVER

$Cross(c_0, S^l(t), PS^l_{best})$ represents that $S^l(t)$ and PS^l_{best} cross, and $Cross(c_1, \delta \oplus Cross(c_0, S^l(t), PS^l_{best}), GS_{best})$ represents that GS_{best} cross with the crossover result of $S^l(t)$ and PS^l_{best}. c_0 and c_1 are learning factors, which represent the number of crossover points with PS^l_{best} and GS_{best} respectively. In this article, crossover points are selected by random strategy. The symbol $\oplus$ means to cross or mutate with a certain probability, as shown below,

$$\delta_0 \oplus Cross(c_0, S^l(t), PS^l_{best}) = \begin{cases} Cross(c_0, S^l(t), PS^l_{best}) & r_0 < \delta_0 \\ S^l(t) & \text{else} \end{cases} \tag{18}$$

$$\delta_1 \oplus Cross(c_1, U^l(t), GS_{best}) = \begin{cases} Cross(c_1, U^l(t), GS_{best}) & r_1 < \delta_1 \\ S^l(t) & \text{else} \end{cases} \tag{19}$$

where δ_0 and δ_1 are constants within the range (0,1). $U^l(t) = \delta_0 \oplus Cross(c_0, S^l(t), PS^l_{best})$, r_0, r_1 are random numbers within the range [0, 1]. when $r_0 < \delta_0$, $S^l(t)$ and PS^l_{best} cross. (19) is the same.

2) MUTATION

$Mutate(c_2, S^l(t))$ represents the mutation operation where c_2 is the number of mutation points. The random strategy is adopted to select c_2 mutation points. In $S^l(t)$, partial values of $r^l(t)$ are randomly changed to value within $[1, |\mathcal{M}|]$, and $z^l(t)$ is randomly changed to the value within [1, 2].

$$\delta_2 \oplus Mutate(c_2, S^l(t)) = \begin{cases} Mutate(c_2, S^l(t)) & r_2 < \delta_2 \\ S^l(t) & else \end{cases} \tag{20}$$

r_2 is the random number within [0, 1].when $r_2 < \delta_2$ the mutation operation is carried out. The specific process of TCA-IPSO algorithm is as follows. The output of the algorithm GS_{best} is the optimal task allocation scheme, which is equivalent to the optimal solution $X = \{x_{ijk}\}$ in problem P1.

In every crossover and mutation, the expectation of the crossover and mutation points are $c_0r_0 + c_1r_1 + c_2r_2$, so the complexity of crossover and mutation is $O(c_0r_0+c_1r_1+c_2r_2)$. The complexity of $G(S)$ and $\overline{T}(S)$ in Fitness function are respectively $O(N + M)$ and $O(\sum_{i=1}^{N} |\mathcal{T}_i|)$. So the complexity of Fitness function is $O(N + M + \sum_{i=1}^{N} |\mathcal{T}_i|)$. The number of particles and iteration are Y and n. The computational complexity of TCA-IPSO algorithm is $O(Y \times n \times (N + M + \sum_{i=1}^{N} |\mathcal{T}_i| + c_0r_0 + c_1r_1 + c_2r_2))$.

Algorithm 1 TCA-IPSO

Input: $\mathcal{N}$, $\mathcal{M}$, $\mathcal{T}_i$,w_{ij},$\hat{t}_i$, C_k, E_k.
Output: GS_{best}

```
Initialize the position of each particle.
for t = 1 to n do
  for l = 1 to Y do
    S^l(t+1) = δ2 ⊕ Mutate(c2, δ1 ⊕ Cross(c1, δ0⊕ Cross(c0, S^l(t), PS^l_best), GS_best))
    Fitness(S) = 1/((1−γ)T̄(S)+γG(S))
    if Fitness(S^l(t)) > Fitness(PS^l_best) then
      PS^l_best = S^l(t)
      if Fitness(PS^l_best) > Fitness(GS_best) then
        GS_best = PS^l_best
      end if
    end if
  end for
end for
```

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. SIMULATION STEPS

In this section, the proposed algorithm is simulated and its performance is verified.

Step 1: Simulate the TCA-IPSO algorithm, then study the influence of different parameters on the convergence effect of the algorithm.

Step 2: Study the impact of UE's number on the cooperation method in TCA-IPSO algorithm.

Step 3: Two intelligent optimization algorithms, PSO and GA, are used to solve P1 problem, and then the convergence effect was compared with TCA-IPSO algorithm.

Step 4: In order to verify the superiority of cooperative computing in TCA-IPSO algorithm, the Benchmark and QBTD method [36] are selected for comparison. In the Benchmark method each task has a fixed resource requirement. And the tasks sent by UEs are only completed by the access point independently. There is no cooperation between the edge nodes. If the access point has insufficient resources, the tasks enter the queue for queuing. In order to reduce the task completion delay, [36] proposes a QoS-based task distribution (QBTD) approach, designed to minimize business completed time delay. Tasks can be executed on any edge node that meets QoS requirements.Finally the task allocation problem is transformed into a mixed integer linear programming problem.

B. SIMULATION SETTINGS

It is assumed that the simulation environment is a square area with side length of 1km. The area contains 10 ENs and 50 UEs, and the positions of EN and UE are randomly

generated in the area. The CPU frequency (GHz) and storage size (GB) of each edge node obey normal distribution, which are $\tilde{N}_1(10, 2\times10^2)$ and $\tilde{N}_2(10^2, 3\times10^2)$ respectively. We set virtual computing unit to be 0.1GHz and virtual storage unit to be 0.5GB. The data transmission rate (KB/s) between two edge nodes follows the normal distribution $\tilde{N}_3(3\times10^3, 10^2)$. The number of subtasks of each UE follows a uniform distribution $U(1, 5)$. The computing delay, the requirements of virtual computing resource unit and virtual storage unit follow poisson distribution. The mean values of poisson distribution are $\lambda_3 = 40, \lambda_1 = 8, \lambda_2 = 10$. Same as [33], [37] the channel gain is expressed as follows: $h = 127 + 30\log d$(d in kilometers), other parameters are set as shown in the Table 2.

TABLE 2. Simulation parameters.

Symbol	Quantity
bandwidth of EN k B_k	20 Mhz
transmission power of UE i p_i	[20,30] dBm
gaussian white noise power σ^2	2×10^{-13} W
subtask input data size d_{ij}	[0.05,0.1] MB
the data size ratio of the output and input λ_{ij}	[0.01,0.1]
population size Y	30
crossover probability δ_0 δ_1	0.6,0.3
mutation probability δ_2	0.1
number of crossover point c_0, c_1	10,8
number of mutation point c_2	5
constraint violation factor γ	0.95

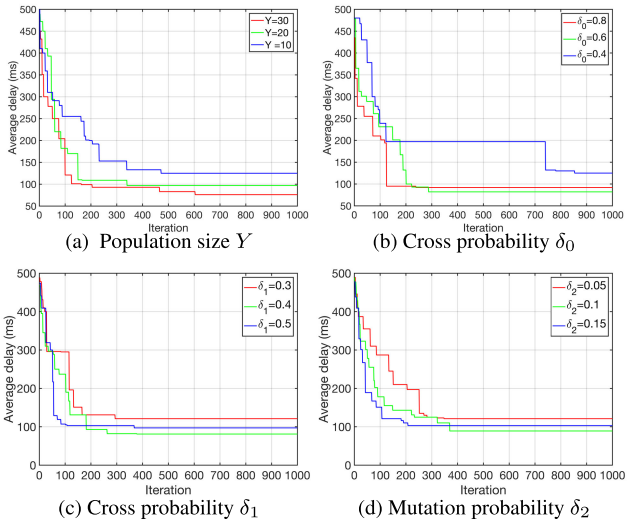
C. SIMULATION RESULTS

1) THE IMPACT OF DIFFERENT PARAMETERS ON THE CONVERGENCE PERFORMANCE OF THE TCA-IPSO ALGORITHM

The TCA-IPSO algorithm includes four important parameters: population size Y, crossover probability with individual optimal particles δ_0, crossover probability with global optimal particles δ_1, and mutation probability δ_2. Different parameters' value has certain effects on search efficiency and solution quality, as shown in Figure 5.

Figure 5 (a) shows the convergence of the TCA-IPSO algorithm under different population numbers. It can be seen that the larger the population size is, the better the quality of the searched solution is and the faster the convergence speed is at the early stage. when $Y = 10$, 20, 30, it converges to 125, 97, and 76 ms respectively. When Y is small, the possibility of falling into a local optimal solution is high due to poor population diversity, and the convergence speed is slow. As the population size increases, the diversity of the population increases. So the probability of finding the optimal solution and the convergence speed increases.

Figure 5 (b) shows the convergence of the TCA-IPSO algorithm under different crossover probabilities δ_0. When δ_0 is 0.4, 0.6, 0.8, it converges to 125, 82, and 92 ms, respectively. The TCA-IPSO algorithm easily converges to the local optimal solution when δ_0 is too large or too small. Smaller crossover probability δ_0 has less impact on population diversity which causes small probability of converging to the global optimum. When δ_0 is larger the probability of local optimum also becomes larger, resulting in poor solution quality.



(a) Population size Y (b) Cross probability δ_0

(c) Cross probability δ_1 (d) Mutation probability δ_2

FIGURE 5. Convergence of TCA-IPSO with different parameters.

Figure 5 (c) shows the convergence of the TCA-IPSO algorithm under different crossover probabilities δ_1. Same as δ_0, when the crossover probability with the global optimal particle δ_1 is too large or too small, the quality of the solution will be reduced and is easy to fall into a local optimum solution. The population diversity will decrease when δ_1 is too small, and the solution will easily fall into local optimization when δ_1 is too large.

Figure 5 (d) shows the convergence of the TCA-IPSO algorithm under different mutation probabilities. When δ_2 is 0.05, 0.1, 0.15, the TCA-IPSO algorithm converges to 121ms, 89ms and 103ms respectively. And it works best at 0.1. The reason is that the small mutation probability has little effect on the improvement of population diversity while large mutation probability leads to the instability of fitness value of particles, which affects the final convergence result. Taking appropriate mutation probability can prevent premature convergence from producing local optimum.

In addition we try to find the best performance of the proposed method under the joint impact of these four parameters. In the experiment the range of the four parameters of Y, δ_0, δ_1 and δ_2 are {10, 20, 30}, {0.4, 0.6, 0.8},{0.3, 0.4, 0.5}, {0.05, 0.1, 0.15}. We performed simulation experiments on 81 cases of four parameter value combinations. The experiment shows that the convergence result is best when $Y = 30, \delta_0 = 0.6, \delta_1 = 0.4, \delta_2 = 0.05$. It converges to 75ms.

2) THE IMPACT OF UE's NUMBER ON RATIO OF THREE COOPERATION METHODS

Figure 6 shows the ratio of the three cooperation methods under different numbers of UE. As the number of UE increases, the ratios of UE using cooperation method 2 and method 3 increase while the ratio of UE using cooperation method 1 decreases. For example, when the number of UE is

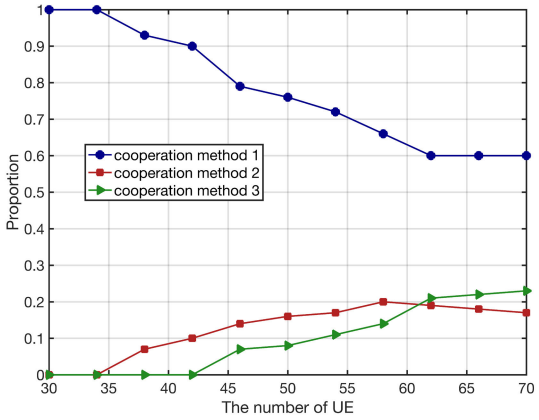


FIGURE 6. Three cooperation methods' ratio with different number of UEs.

50, the ratios of UE number in cooperation method1,2,3 are 0.76, 0.16 and 0.8. respectively. When the number of UE is small single EN can meet the requirement. As the number of UE increases, due to limited resources some EN need to cooperate with their neighbor ENs to complete tasks to meet the business requirements. When it continues to increase, part of EN's remaining resources are consumed. In order to meet the business requirements and reduce the business completion delay, cooperation 3 method is adopted.

3) COMPARISON OF CONVERGENCE PERFORMANCE OF DIFFERENT INTELLIGENT OPTIMIZATION ALGORITHMS

PSO and GA were used to solve the P1 problem, and the convergence effect was compared with the proposed TCA-IPSO algorithm, as shown in Figure 7.

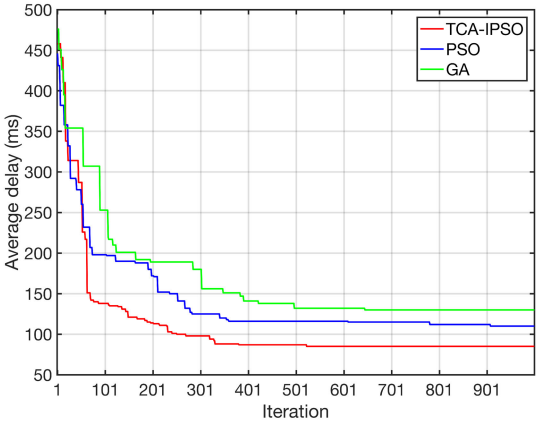


FIGURE 7. Convergence performance of different intelligent optimization algorithms.

We can see that the performance of TCA-IPSO is the best, and the convergence result is 74ms, while the convergence results of PSO and GA are 111ms and 131ms. GA updates chromosome through selection, crossover and mutation to improve population diversity. However, memory is not preserved in iterations, and previous knowledge is destroyed with population change. It is easy to converge to the local optimal solution. PSO algorithm preserves historical memory, and updates particles by using the sharing mechanism of historical information of individuals in the group. However, the method of update strategy is not good for discrete optimization problems, which affects the quality of solutions. Simultaneously PSO algorithm prematurely converges, and the local optimization ability is poor. TCA-IPSO applied crossover and mutation strategy to improve PSO. It enhances the renewal ability of particle swarm and jumping out of the local optimum. It improves the problem of premature convergence and local optimization.

4) COMPARISON OF DIFFERENT TASK ALLOCATION METHODS

TCA-IPSO is compared with Benchmark and QBTD method. The average delay of the three methods under different UE's number, CPU frequency, storage size and UE distribution unbalance degree were compared respectively.

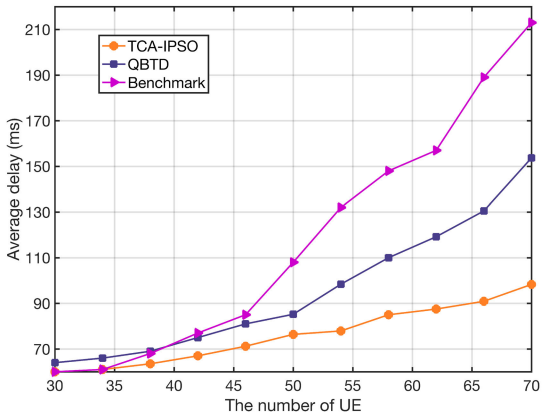


FIGURE 8. Average delay of task allocation method with different number of UE.

Figure 8 shows the average delay comparison of three task allocation mechanisms, TCA-IPSO, Benchmark and QBTD under different UE's numbers. With the increase of the number of UE, the average delay increases gradually, and TCA-IPSO has the best effect. In Benchmark method, tasks can only be completed on access point EN. If the access point has insufficient free resources, the task will enter the queue for queuing. As the number of UE increases, it is difficult for a single EN to meet the all task requirements at the same time. Therefore, tasks need to be queued on EN for completion, resulting in an extremely increased average completion time. QBTD method distributes tasks among edge nodes to avoid excessive load on some ENs, so it reduces queuing delay. TCA-IPSO segments the task to subtasks and distributes them to two ENs. Therefore, with the increase of the number of UE, the average delay increases slowly. When UE's number is 50, compared with Benchmark and QBTD mechanism, the average delay decreases by 29.3% and 10.3% respectively, and when UE's number is 70, it decreases by 53.8% and 36.0% respectively.

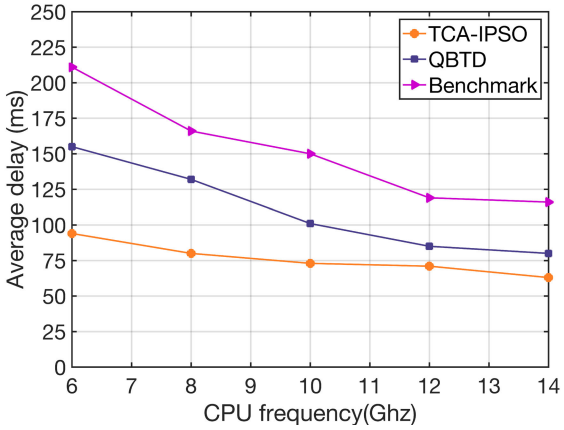


FIGURE 9. Average delay of task allocation method with different CPU frequencies.

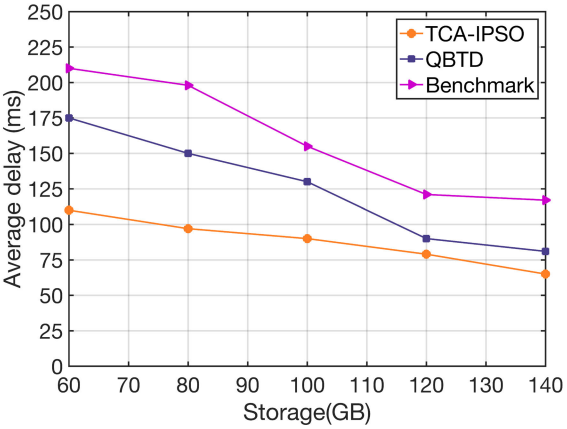


FIGURE 10. Average delay of task allocation method with different storage size.

Figure 9 and Figure 10 respectively show the average delay of the three task allocation mechanisms under different CPU frequency and storage size. It can be seen that the average delay of the three methods gradually decreases with the increase of resources. The average delay of TCA-IPSO has been kept to the minimum, because when EN load is too high and the remaining resources are unable to meet the business requirement, some subtasks are diverted to other idle EN to meet the business requirement through cooperation. In Benchmark method, for the heavily loaded EN the task can only queue in the access point's queue until there is enough free resources. With the increase of computing and storage resources of edge node, some tasks in the queue can be directly completed. It avoids part of waiting delays in the queue. Therefore, the average completion delay is significantly reduced. However, considering the different load degree in the ENs, some tasks still need to wait in the queue on the heavily loaded EN, so the effect is worse than that of TCA-IPSO and QBTD. In QBTD, the tasks cannot be divided and need to be assigned to other nodes, which brings more communication delay.

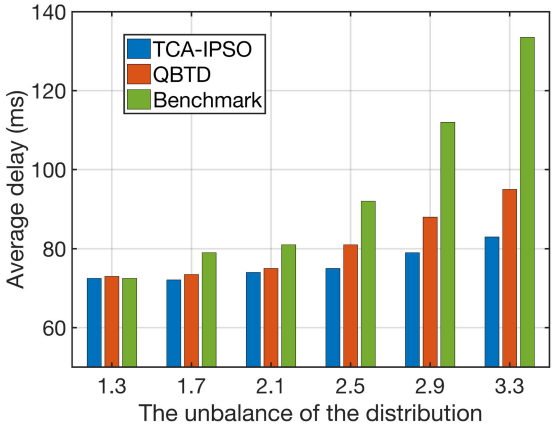


FIGURE 11. Average delay of task allocation whit different UE distribution unbalance degree.

Figure 11 shows the average delay of the three task allocation mechanisms under different UE distribution unbalance degree. The distribution unbalance degree is represented by the variance σ of number of UEs associated with EN, $\sigma = \sqrt{\frac{1}{M}\sum_{k=1}^{M}(|\mathcal{N}_k - \mu|)^2}, \mu = \frac{\sum_{k=1}^{M}(|\mathcal{N}_k|)}{M}$. It can be seen that the average delay of the three task allocation mechanisms increases when σ increases. The average delay of TCA-IPSO is lower than QBTD and Benchmark. For example,when $\sigma = 2.5$ the average delay of TCA-IPSO, QBTD and Benchmark is 75ms, 81ms and 93ms respectively. Benchmark has the highest average delay because the higher load EN does not divert tasks to other EN. It causes tasks to queue and task completion delay increases. QBTD mechanism performs task assignment in the edge network, avoiding the phenomenon of task queuing and reducing the task completion delay. TCA-IPSO divides the task into subtasks, which are executed in parallel on two nodes to further reduce the average delay.

VI. CONCLUSION

The resources of edge nodes in the power IoT scene are limited, so it is necessary to meet all the increasing business processing demands timely through the cooperation of edge nodes. For the issue that edge node resource is limited and business requirements cannot be met at the same time in the power IoT scene, task allocation mechanism based on cooperative edge computing is proposed. First of all, task allocation model based on cooperation of two edge nodes is build. In the model business request, resource and delay are analyzed. According to whether the access point participates in cooperation the three cooperation methods are proposed. The model aims to minimize the average completion delay under the constraints of business requirements, and it describes the problem as an integer nonlinear programming problem. Then we propose TCA-IPSO algorithm, which improves the particle swarm optimization algorithm by applying the crossover and mutation strategy in the genetic algorithm to

solve the problem. Finally the simulation results show that the TCA-IPSO algorithm reduces the average task completion delay by 53.8% and 36.0% compared to the benchmark and QBTD algorithm. TCA-IPSO algorithm can be applied in power IoT scene with massive business terminals and limited edge node resources to solve the problems of insufficient edge node resources and business processing capacity.

The TCA-IPSO algorithm proposed in this article is a heuristic algorithm. Due to the uncertainty of the heuristic algorithm, its complexity cannot be accurately measured, and the efficiency of solving the task allocation problem in the power IoT still can be improved. In the future, we will explore new hybrid algorithm and parallel optimization algorithm to further improve the algorithm search efficiency.

REFERENCES

[1] B. Huang, Y. Li, H. Zhang, and Q. Sun, "Distributed optimal co-multi-microgrids energy management for energy Internet," *IEEE/CAA J. Automatica Sinica*, vol. 3, no. 4, pp. 357–364, Oct. 2016.

[2] Y. Zhang, K. Liang, S. Zhang, and Y. He, "Applications of edge computing in PIoT," in *Proc. IEEE Conf. Energy Internet Energy Syst. Integr. (EI2)*, Nov. 2017, pp. 1–4.

[3] X. Niu, S. Shao, C. Xin, J. Zhou, S. Guo, X. Chen, and F. Qi, "Workload allocation mechanism for minimum service delay in edge computing-based power Internet of Things," *IEEE Access*, vol. 7, pp. 83771–83784, 2019.

[4] 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.

[5] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 2017.

[6] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct. 2017.

[7] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4188–4200, Jun. 2019.

[8] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Communication-aware scheduling of serial tasks for dispersed computing," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1330–1343, Aug. 2019.

[9] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3512–3524, Apr. 2019.

[10] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: An ADMM framework," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2763–2776, Mar. 2019.

[11] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3056–3069, Nov. 2017.

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

[13] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2146–2153, Jun. 2018.

[14] Y. Li, G. Xu, J. Ge, X. Fu, and P. Liu, "Communication and computation cooperation in wireless network for mobile edge computing," *IEEE Access*, vol. 7, pp. 106260–106274, 2019.

[15] H. Ju and R. Zhang, "User cooperation in wireless powered communication networks," in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 1430–1435.

[16] Y. Dong, S. Guo, J. Liu, and Y. Yang, "Energy-efficient fair cooperation fog computing in mobile edge networks for smart city," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7543–7554, Oct. 2019.

[17] W. Fan, Y. Liu, B. Tang, F. Wu, and Z. Wang, "Computation offloading based on cooperations of mobile edge computing-enabled base stations," *IEEE Access*, vol. 6, pp. 22622–22633, 2018.

[18] M. Safar, I. Ahmad, and A. Al-Yatama, "Energy-aware computation offloading in wearable computing," in *Proc. Int. Conf. Comput. Appl. (ICCA)*, Sep. 2017, pp. 266–278.

[19] R. Beraldi, A. Mtibaa, and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, May 2017, pp. 94–100.

[20] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Trans. Commun.*, vol. 66, no. 4, pp. 1594–1608, Apr. 2018.

[21] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia IoT systems," *IEEE Trans. Multimedia*, vol. 20, no. 5, pp. 1126–1139, May 2018.

[22] S. Zhu, L. Gui, J. Chen, Q. Zhang, and N. Zhang, "Cooperative computation offloading for UAVs: A joint radio and computing resource allocation approach," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, San Francisco, CA, USA, Jul. 2018, pp. 74–79, doi: 10.1109/EDGE.2018.00017.

[23] C. Gong, F. Lin, X. Gong, and Y. Lu, "Intelligent cooperative edge computing in the Internet of Things," *IEEE Internet Things J.*, early access, Apr. 8, 2020, doi: 10.1109/JIOT.2020.2986015.

[24] Y. Liu, K. Xiong, Q. Ni, P. Fan, and K. B. Letaief, "UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, CPU control, and trajectory optimization," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2777–2790, Apr. 2020, doi: 10.1109/JIOT.2019.2958975.

[25] X. Chen, Z. Zhou, W. Wu, D. Wu, and J. Zhang, "Socially-motivated cooperative mobile edge computing," *IEEE Netw.*, vol. 32, no. 6, pp. 177–183, Nov. 2018, doi: 10.1109/MNET.2018.1700354.

[26] J. Wu, Z. Cao, Y. Zhang, and X. Zhang, "Edge-cloud collaborative computation offloading model based on improved partical swarm optimization in MEC," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Tianjin, China, Dec. 2019, pp. 959–962.

[27] C. Chen, L. Chen, L. Liu, S. He, X. Yuan, D. Lan, and Z. Chen, "Delay-optimized V2 V-based computation offloading in urban vehicular edge computing and networks," *IEEE Access*, vol. 8, pp. 18863–18873, 2020.

[28] B. Y. Cheng, H. Lu, X. Xu, and W. Shen, "Improved local-search-based chaotic discrete particle swarm optimization algorithm for solving traveling salesman problem," *J. Comput. Appl.*, vol. 36, no. 1, pp. 138–142, 2016.

[29] Y. Guangming, Z. Yunfei, D. Chengjun, and Z. Peng, "AGV path planning based on improved genetic algorithm," *J. Beijing Union Univ.*, vol. 32, no. 1, pp. 65–69, 2018.

[30] 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.

[31] R. Miao, W. Yan, and S.-R. Li, "Novel hybrid GA based on position displacement idea of PSO and its application," *Jisuanji Gongcheng yu Yingyong(Comput. Eng. Appl.)*, vol. 43, no. 15, pp. 37–40, 2007.

[32] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoT–edge–cloud computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2759–2774, Dec. 2019.

[33] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018, doi: 10.1109/JSAC.2018.2815360.

[34] X. Chen and J. Zhang, "When D2D meets cloud: Hybrid mobile task offloadings in fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6, doi: 10.1109/ICC.2017.7996590.

[35] 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, doi: 10.1109/JIOT.2018.2838022.

[36] Y. Song, S. S. Yau, R. Yu, X. Zhang, and G. Xue, "An approach to QoS-based task distribution in edge computing networks for IoT applications," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Honolulu, HI, USA, Jun. 2017, pp. 32–39, doi: 10.1109/IEEE.EDGE.2017.50.

[37] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017, doi: 10.1109/JSAC.2017.2760160.

QIANJUN WANG is currently pursuing the M.E. degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His current research interests include the power Internet of Things and edge computing.

SUJIE SHAO (Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015. He is currently a Lecturer with the Beijing University of Posts and Telecommunications. His research interests include edge computing, the Internet of Things, smart grids, and communication network management.

SHAOYONG GUO received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2013. He is currently an Associate Professor with the Beijing University of Posts and Telecommunications. His research interests include blockchain, the Internet of Things, ubiquitous networks, and smart grids.

XUESONG QIU (Senior Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2000. He is currently a Professor and a Ph.D. Supervisor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. He has authored about 100 SCI/EI indexed articles. He presides over a series of key research projects on network and service management, including the projects supported by the National Natural Science Foundation and the National High-Tech Research and Development Program of China.

ZHILI WANG is currently an Associate Professor with the Beijing University of Posts and Telecommunications, where he was engaged in research and standardization work in communication networks and computer science, and technology. He wrote more than eight ITU-T international standards. His main research interests include network management, communications software, and interface testing. He won one National Science and Technology Progress Award. He serves as the Working Party 2 Chairman for the ITU-T Study Group 2.