IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 16, 2019, accepted October 1, 2019, date of publication October 11, 2019, date of current version October 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2946683

An Energy-Efficient Off-Loading Scheme for Low Latency in Collaborative Edge Computing

JIN WANG[1,2], WENBING WU[1], ZHUOFAN LIAO[1], (Member, IEEE), ARUN KUMAR SANGAIAH[3], (Member, IEEE), AND R. SIMON SHERRATT[4], (Fellow, IEEE)

[1]Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410000, China
[2] School of Information Science and Engineering, Fujian University of Technology, Fuzhou 350118, China
[3] School of Computing Science and Engineering, Vellore Institute of Technology (VIT), Vellore 632014, India
[4]School of Systems Engineering, University of Reading, Reading RG6 6AY, U.K.

Corresponding author: Zhuofan Liao (zfliao@csust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772454, Grant 61811530332, and Grant 61811540410, and in part by the Degree & Postgraduate Education Reform Project of Hunan Province under Grant 2019JGZD057.

ABSTRACT Mobile terminal users applications, such as smartphones or laptops, have frequent computational task demanding but limited battery power. Edge computing is introduced to offload terminals' tasks to meet the quality of service requirements such as low delay and energy consumption. By offloading computation tasks, edge servers can enable terminals to collaboratively run the highly demanding applications in acceptable delay requirements. However, existing schemes barely consider the characteristics of the edge server, which leads to random assignment of tasks among servers and big tasks with high computational intensity (named as ''big task'') may be assigned to servers with low ability. In this paper, a task is divided into several subtasks and subtasks are offloaded according to characteristics of edge servers, such as transmission distance and central processing unit (CPU) capacity. With this multi-subtasks-to-multi-servers model, an adaptive offloading scheme based on Hungarian algorithm is proposed with low complexity. Extensive simulations are conducted to show the efficiency of the scheme on reducing the offloading latency with low energy consumption.

INDEX TERMS Latency, energy, offloading, edge computing.

I. INTRODUCTION

Mobile terminal devices are connected through the internet to accomplish many different applications and services, such as smartphones, laptops, sensors, machines, and vehicles, etc[1]. To extract valuable information from the huge amount of users' data, local computation with terminal devices are no longer provide demanding quality of services such as low latency and energy consumption[2], [3], especially for video image stream data processing[4]–[6]. In-vehicle networks, tasks with high latency sensitivity require lower processing time. Otherwise, message propagation among vehicles may fail [7]. Therefore, light-weighted servers are deployed on the edge around terminals to bring computation and storage resource from the centralized cloud (CC), which is called as Mobile Edge Computing (MEC) [8]. Tasks generated by terminals can be offloaded and processed on edge servers [9]–[10] instead of being transferred to CC with large delay, and tasks or applications can effectively meet the delay requirements [11]–[13]. As privacy and security become more important in our daily life [14]–[17], a low delay would be particularly important in privacy and security issues for mobile edge computing systems.

Many researches are devoted to reduce the offloading time and energy consumption for edge computing. More and more researchers considered the difference of tasks, such as computation-intensive task, delay-sensitive task, etc. For these scenarios, relationships between servers are also considered, such as the hierarchical servers, called as collaborative edge computing (CEC). CEC allows multiple servers to collaboratively offload different type of tasks to efficiently reduce time and energy consumption.

However, previous work only focused on how to offload different types of task. The difference of computation

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

intensity caused by different types of demand are rarely taken into account in existing designs. Meanwhile, characteristics of servers, such as physical distance and CPU capacity, are not considered. If tasks with high computational intensity which we name as ''big tasks'' is assigned to servers with low ability or long task processing queue, the delay of offloading will be very large and the whole offloading process is choked.

In this paper, we focus on the relationship between tasks and servers, characteristics of edge servers, and a task can be divided into several subtasks to be offloaded to different servers. An adaptable offloading scheme based on Hungarian algorithm is designed to allocate subtasks to edge servers to reduce offloading latency and energy consumption. The main contributions of this paper are as follows:

1) A collaborative task offloading model is proposed in edge computing system.
2) We formulate the task offloading problem and design a distributed task offloading scheme to solve this problem.
3) Extensive simulations are designed to evaluate performance.

The rest of this article is organized as follows. Section 2 introduces related works. Section 3 introduces the offloading model. The offloading scheme is presented in Section 4. In Section 5, simulation results are illustrated and discussed. And Section 6 concludes the paper.

II. RELATED WORK

In order to get low latency and energy consumption in edge computing, offloading is considered and widely adopted. For efficient offloading, existing offloading method can be summarized into three categories: partial offloading, fully offloading and preference offloading.

A. PARTIAL OFFLOADING

Considering that the storage and computation capacity of servers is limited, to guarantee the each task can be executed in one time slot, a task is divided into two or more parts, one part is processed at the local server, and the others is processed remotely [18]–[20]. By dividing task reasonably, the execution time or energy of the whole task can be effectively reduced. Shurman and Aljarah [21] proposed a collaborative method of distributing marginal resources for pre-partitioned application modules, which maximized the utilization of edge resources. This method leaded to less latency and less traffic over the network compared to executing modules on the cloud, which provides users with faster service delivery and reduced core network traffic. Wu *et al.* [22] proposed a dynamic task partitioning algorithm that can determine the optimal allocation of tasks performed locally or remotely. A weighted resource consumption map (WRCM) was constructed and a Minimum Cost Offloading Partition Algorithm (MCOP) is further proposed. The adaptive partitioning was briefly analyzed by program profiler, network profiler and network profiler. The algorithm can effectively reduce network overhead and can be applied when the network changes. He *et al.* [23] studied how to improve the computation capacity of cellular networks. A device-to-device mobile edge computing (D2D-MEC) technology is proposed and a mixed integer nonlinear problem is formed. This problem can be divided into two sub-problems, the first sub-problem minimizing the need of computation resources for a given edge D2D pair, the second sub-problem was based on the first sub-problem solution, maximizing the number of devices the system can support. By solving these two sub-problems, the computational capacity was effectively improved.

B. FULLY OFFLOADING

Fully offloading means the whole task offloaded to a device or a server, and the device or the server can process the task in a reasonable latency. Chen *et al.* [24] analyzed the multi-channel and multi-user computing offloading decision problem in edge computing system. They proved that this problem will always admits a Nash equilibrium, and designed a distributed computing offloading algorithm to achieve Nash equilibrium and effectively reduced time consumption by effectively solving problems. Wei *et al.* [25] investigated the scenario of multiple cell phone upload tasks to an MEC server and the challenge of allocating limited server resources and wireless channels between devices. A Select Maximum Saved Energy First (SMSEF) algorithm was proposed to solve the energy optimization problem of mobile devices with dividable tasks. Xing *et al.* [26] studied a new D2D multi-assistant MEC system. The authors employed a time division multiple access (TDMA) transport protocol. In this protocol, local users offloaded tasks to multiple assistants and download results from them at predetermined orthogonal intervals, which reduced computational latency, computational frequency of task execution, and algorithm complexity. Zhang *et al.* [27] proposed a contract-based computation resource allocation method, which improved the utility of the vehicle terminal in mobile edge computing. Xu *et al.* [28] explored a resource allocation method in the Internet of things (IoT) environment. A model named Zenith is proposed to establish an auction-based resource allocation contract, and a task scheduling model is developed according to specific tasks. Kim *et al.* [29] studied the resource management of mobile devices in a tradeoff environment. A series of short-term goals are obtained by using Lyapunov optimization technology, and an optimization algorithm was proposed. Chen *et al.* [30] took the task offloading problem as a stochastic optimization problem, and used a stochastic optimization technology to transform a random problem into a deterministic optimization problem. Zhang *et al.* [31] studied the trade-off relationship between energy consumption and time consumption in edge computing. An energy-aware offloading scheme was proposed to jointly optimize communication and computing resources with limited energy and delay sensitivity for time and energy reducing

C. PREFERENCE OFFLOADING

Some researches focused on offloading tasks according to some preferences. Jiang *et al.* [32] studied the relationship

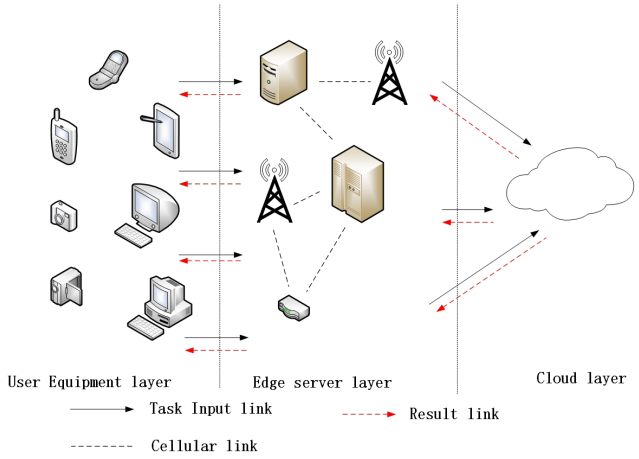


FIGURE 1. The edge computing system model.

between content popularity and user preference and task offloading in edge computing. The user popularity is predicted in the online phase and the user's preferences are learned in the offline phase. Through "Follow The (Proximally) Regularized Leader" (FTRL-Proximal) algorithm and Online Gradient Descent (OGD) algorithm, the cache-hit rate is improved. This method can reduce the computational complexity and optimize the edge cache problem.

In the paper, we focus on partial task assignment in edge computing system, but differ from above works. The above work about partial task assignment did not consider that a task can be divided into several subtasks and the synergy of servers. In this scheme, local server divides big tasks into subtasks and assigns them to neighbor servers based on Hungarian algorithm is designed, which take the characteristics of servers into account.

III. SYSTEM MODEL

Considering a mobile edge system consists of M user equipment and N edge servers. Edge servers connect with each other by cellular link. For each user, the nearest server that can provide task offloading service is called as the *local server*, while servers that are one-hop far away from the local server are called as *neighbor servers* of the local server. Assumed that each user generate only one task at a time slot and the i*th* task is described by a triple tuple $< D_i, C_i, T_i^{tolerate} >, i = \{1, 2, \cdots, M\}$, where D_i is the data size, C_i is the CPU cycle required to successfully process the i*th* task and $T_i^{tolerate}$ is the time tolerance of the i*th* task. Each task can be divided into several subtasks, especially for video and image streaming tasks. Then subtasks can be distributed to different servers to process [22]. This assumption also works for interdependent tasks, because such tasks can be partially offloading. One part of subtasks are processed locally and the rest part are processed remotely [20], [22], [33]–[35].

Servers will also provide computation offloading service to users and share computation resources among other edge servers to solve the problem of limited capacities of user equipment and other edge servers. Servers will periodically exchange their own information with their neighbors through cellular link. The servers' information is presented as a triple $< f_j, d_j, w_j >, j = \{1, 2, \cdots, N\}$ to describes the characteristics of each server, where f_j is the CPU capacity of the server j, d_j is the distance, and w_j is the waiting time if subtask need to be executed in server j.

A local server is considered being connected with K neighbor servers, and it will divide each task into $(k+1)$ subtasks. One subtask will be left on the local server and the others will be assigned to the k neighbor servers. The information of each subtasks is presented as a triple $< \mathrm{D_{i,u}}, \mathrm{C_{i,u}}, T_{i,u}^{tolerate} >$, where $\mathrm{D_{i,u}}$ is the subtask size, $\mathrm{C_{i,u}}$ is the required CPU cycle and $T_{i,u}^{tolerate}$ is the time tolerance of subtask.

A. TIME CONSUMPTION

In practical researches, the latency mainly consists of three parts: transmission latency, computation latency and queuing latency. In this work, transmission latency is the time spent on transferring a subtask from the local server to a neighboring one. Computation latency is the subtask processing time on the server. Queuing latency is the waiting time that a subtask costs on a server.

In this work, the orthogonal multiple access (OMA) based communication technology is adopted during the communication between edge servers. OMA based communication is the communication technology in present fourth generation (4G) and widely used in daily life since non-orthogonal multiple access (NOMA) based communication technology of the fifth Generation (5G) technology communication technology is still in a research and development stage. Thus the signal interference during transmission can be ignored [34]. According to Shannon formula [35], the system date rate between local server and neighbor server j is given as follows.

$$\mathrm{r}\,(u, j) = B \log\left(1 + \frac{p_{u,j}^{tra} h_{u,j}}{N_0}\right) \tag{1}$$

, where B is the network bandwidth,, N_0 is the background noise power, $p_{u,j}^{tra}$ is the transmission power, $h_{u,j}$ is the channel power gain between subtask u and neighbor server j. Then, the transmission latency for the subtask u to edge server j can be presented as follows:

$$T_{u,j}^{tra} = \frac{D_{i,u}}{\mathrm{r}\,(u, j)} \tag{2}$$

If a subtask is processed on the local sever, the transmission delay zero. Noting that the download transmission delay and packet loss of the subtask is not considered because the size of subtask shrinks sharply after processed [24].

The computation latency when subtask u is executed in edge server j is depending on the server's computing capacity and can be described as follows [20], [29], [33]–[36].

$$T_{u,j}^{com} = \frac{C_{i,u}}{f_j} \tag{3}$$

By adding the three types of latency, the total latency $T_{u,j}^{total}$ of the subtask u allocating to the server j is constructed

as follows:

$$T_{u,j}^{total} = \begin{cases} T_{\mathrm{u,j}}^{com} + w_j, & \textit{if subtask u executed} \\ & \textit{in local server} \\ T_{\mathrm{u,j}}^{com} + T_{u,j}^{tra} + w_j, & \textit{otherwise} \end{cases} \tag{4}$$

where w_j is the waiting time when subtask executed in edge server j. As mentioned above, one subtask will be left in local server for executing to make fully use of limited resource in local server. Thus $T_{u,j}^{tra} = 0$ if subtask executed in local server due to subtask doesn't need to be transferred to neighbor server.

B. ENERGY CONSUMPTION

In this work, we focus on the energy consumption which is the major concern that needs to be addressed. Energy consumption is mainly composed of two parts: the consumption of energy generated in the procedure of transmission; the consumption of energy generated in the procedure of server processing tasks.

On the one hand, the energy consumption in the procedure of transmission is as follows:

$$E_{u,j}^{tra} = \frac{D_{i,u} p_{u,j}^{tra}}{\mathrm{r}\,(u,j)} \tag{5}$$

On the other hand, the energy consumption in the procedure of processing tasks in the server is as follows[20]:

$$E_{u,j}^{com} = \frac{C_{i,u} p_j^{com}}{f_j}, \tag{6}$$

where p_j^{com} is the energy consumption of per second of server j. By adding the two type of energy consumption, the total energy consumption is obtained as follows:

$$E_{u,j}^{total} = \begin{cases} E_{u,j}^{com}, & \textit{if subtask u executed} \\ & \textit{in local server} \\ E_{u,j}^{com} + E_{u,j}^{tra}, & \textit{otherwise} \end{cases} \tag{7}$$

If the subtask is executed in local server, then $E_{u,j}^{tra} = 0$ because the subtask does not need to be transferred to neighbor server.

$$\mathrm{G}_i^{energy} = \begin{array}{c} \begin{array}{ccccccc} \mathrm{LS} & \mathrm{NS\ 1} & \cdots & \mathrm{NS}\ j & \cdots & \mathrm{NS}\ K & \end{array} \\ \begin{bmatrix} E_{1,0}^{total} & E_{1,1}^{total} & \cdots & E_{1,j}^{total} & \cdots & E_{1,K}^{total} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ E_{u,0}^{total} & E_{u,1}^{total} & \cdots & E_{u,j}^{total} & \cdots & E_{u,K}^{total} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ E_{K+1,0}^{total} & E_{K+1,1}^{total} & \cdots & E_{K+1,j}^{total} & \cdots & E_{K+1,K}^{total} \end{bmatrix} \end{array} \begin{array}{c} \\ \text{subtask 1} \\ \vdots \\ \text{subtask } u \\ \vdots \\ \text{subtask } K+1 \end{array} \tag{8}$$

A subtask will have different energy consumption when it is executed on different edge servers which are different in computation capacity, distance, transmission power, and CPU cost. The energy consumption matrix G_i^{energy} for subtasks is constructed as a matrix shown in expression (8). $E_{u,j}^{total}$ stands for the energy consumption when subtask u executed on edge server j. The row is the energy consumption of subtask if it is offloading on corresponding edge servers where the local server is denoted as LS, the j*th* neighbor server is denoted as $\mathrm{NS_j}$. And the column is the energy consumption of different subtasks on the same edge server.

C. OPTIMIZATION GOAL

A vector $X = (x_1, \ldots, x_u, \ldots, x_{k+1})$ is used to indicate which server the subtask u of task i is assigned to, $x_u = 0$ indicates subtask $u$is executed on local server, $x_u = j$ indicates subtask u is executed on neighbor server j. Then we can formulate the optimization goal as follows

$$\min_X \sum_u E_{u,j}^{total} \tag{9a}$$

$$\text{s.t.} \sum_u T_u^{total} < T_i^{tolerate} \tag{9b}$$

$$x_u \le k \tag{9c}$$

$$\frac{p_u^{tra} p_{u,j}^{gain}}{N_0} \ge \Gamma \tag{9d}$$

where constraints (9b) indicates that the sum time consumption of all subtasks should be less than task time tolerance. Constraints (9c) indicates that the subtask can be only assigned among k neighbor servers and the local server. Constraints (9d) indicates that the Signal to Noise Ratio (SNR) must be higher than a threshold value to ensure successful transmission.

IV. ALGORUTHM DESIGN

In this section, we propose an energy overhead optimize task offloading scheme under multiple subtasks and multiple edge servers. The task offloading scheme consists of two phases, (1) the task division phase and (2) the subtask assignment phase. A task is divided according to the number of neighbor servers in the first phase of the algorithm, then we focus on the assignment strategy in the second phase. Next, a specific illustration is given on how a local server assign subtasks to its neighbor servers.

A. ILLUSTRATION OF ALGORITHM

As shown in Figure 2, when a task arrives at the local server, it may be divided locally into several subtasks, which depends on the computational intensity of the task and the computer capabilities of the local server. If the local server is busy in computation or has a long task-processing queue, subtasks will be assigned to other selected neighbor servers.

In task division phase, the number of subtasks is determined by the number of neighbor servers to make full use of the resources. At least one subtask is kept on the local server to take full advantage of the resources of the local server.

After the task is divided into subtasks on the local server, the subtasks would then be assigned to the neighbor servers considering several realistic factors of the server including local server resource limits, physical distance, server CPU capacity which goes to the second assignment phase.

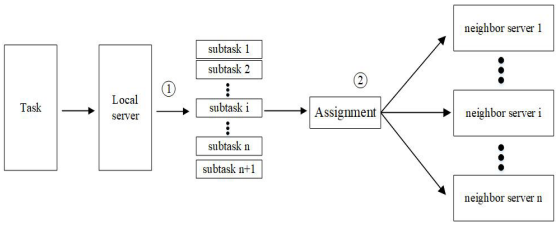


FIGURE 2. Illustration of the proposed algorithm.

Noting that the subtask assigned to neighbor servers have different transmission time and energy consumption. The physical distance and transmission power between different neighbor servers in the real world are different, which makes the transmission time consumption and energy consumption different. For example, the distance between the local server and the selected neighbor server may be very long, but the computing capacity of the neighbor server may be very high [28]. The distance between the local and neighbor servers may be short, but the computing capacity of the neighbor server may be low. Therefore, it is necessary to select neighbor servers elaborately to effectively reduce the energy consumption of the entire tasks.

B. SOLUTION

Since different subtasks are processed on different servers with different time and energy consumption, it is critical to assign which subtask to which server with minimum energy. After the first phase, two sets are produced, one is subtasks set, and the other is neighbor server set. To determine the best server for each subtask, a Multi-subTasks-to-Multi-Servers offloading scheme (MTMS) is proposed based on the Hungarian algorithm to assigning subtasks during the subtasks assignment phase.

Because edge servers broadcast the information about themselves, include CPU frequency, position and some network information and so on, each local server will maintain and update information about its neighbor servers. After the first division phrase, the local server decides to assign tasks to neighbor servers. The local server makes a calculation about transmission time and execution time when subtasks processed in servers, then forms an energy overhead matrix by add the transmission energy and the computation energy about subtasks processed in neighbor servers. After that, Hungarian algorithm is used to make the perfect decision about subtasks assignment. Finally, local server assigns subtasks to those servers. The detailed description is presented in Algorithm 1 as follows.

C. STABILITY AND COMPLEXITY ANALYSIS

Given a subtask set and a neighbor server set, the MTMS algorithm is stable if can each subtask can be successfully be offloaded by a server. Because we assume that a server accepts only one subtask in a time slot, and the number of subtasks to be assigned is equal to the number of neighbor servers plus one (local server). According to the Hungarian

Algorithm 1 The Multi-SubTasks-to-Multi-Servers Offloading Scheme (MTMS)

Input: server number, servers' position, servers' CPU capacity, task number, tasks' size, tasks' CPU cycle
Output:task assignment decision.
Initialization with task set $\{t_i\}$ and server set $\{s_j\}$, $i \in \{1, \ldots M\}$,$j \in \{1, \ldots N\}$ Each server broadcasts its information, including CPU frequency, position and so on.
for each task ti, $i \in \{1, \ldots M\}$ **do**
  // Task division phase
  splitting the task ti it got into (k + 1) subtasks in local server, k is the number of neighbor servers, $k < N$.
  // subtask assignment phase.
    calculating the transmission time and computation time of each subtask to each neighbor servers.
  calculating the transmission energy and computation energy of each subtask if it would be assigned to corresponding neighbor server.
  generating the energy overhead matrix A by adding the transmission and computation energy.
  while exist subtask not assigned **do**
    choose a subtask *u* from subtask set
    choose the edge server *w* which has minimum energy consumption when execute subtask *u* according to the overhead matrix A
    if w hasn't accept any subtask **then**
      assign subtask *u* to *w*
    else//*w* has accept subtask *u*'
      if the energy consumption of subtask *u*' lower than that of subtask *u* in *w* **then**
        subtask u is not assigned to w
      else
        w discord subtask *u*' and accept subtask *u*
        subtask *u*' back to subtask set
  end while
  assigning subtasks to servers according to the subtask assignment decision.
end for

algorithm, each subtask will be assignment to at least one neighbor server.

The other major advantage of the MTMS algorithm is the low-degree polynomial complexity [37]. From a practical point of view, Hungarian algorithm has produced solutions to many industrial problems that were hitherto intractable. A task is divided into $(n + 1)$ subtasks, and each subtask need to compute the transmission and execution energy on every servers, then an overhead matrix is formed by adding transmission and execution energy, so the time complexity is $O(n^2)$ in constructing overhead matrix. The time complexity is $O(n^3)$ in decision making by using Hungarian algorithm. Thus the time complexity of MTMS is $O(n^3)$, and n is the number of subtasks.

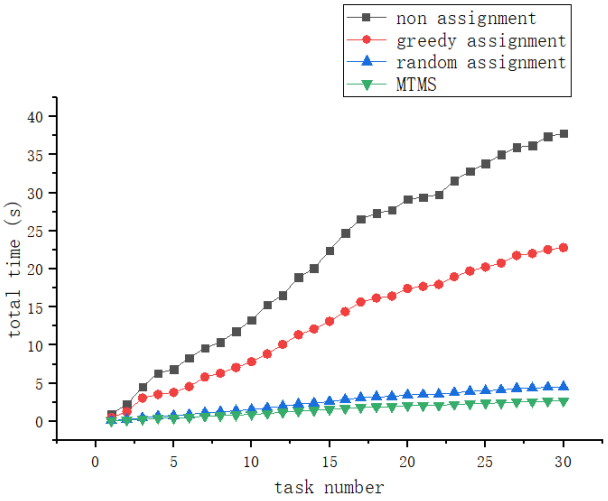
V. EVALUATE THE PERFORMANCE

In this section, the performance of the multi-server task offloading scheme is investigated. We compare MTMS with a Non-assignment scheme, Greedy assignment scheme [23] and Random assignment to evaluate the performance.

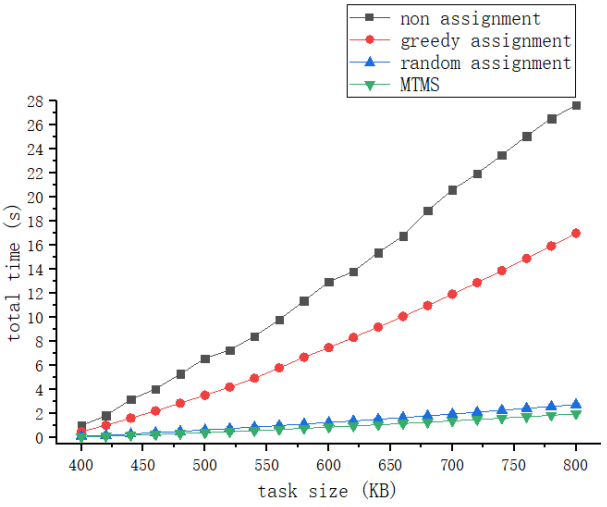
1) Non-assignment scheme: tasks are stored and processed on the local server.
2) Greedy assignment scheme [25]: when the task is not processed on the local server, the neighboring server with the largest CPU capacity is chosen to be the server that the whole task is offloaded to.
3) Random assignment scheme: a task is divided into several subtasks and randomly assigned to multiple neighbor servers for processing. This is the most common and widely used task unloading method in industry at present.

We simulated the experiment on a windows computer, which contains a dual-core CPU, 4 gigabytes of memory and 200 gigabytes of external memory. Here, 10 servers are set up in the range of 50m × 50m. Without losing generality, we refer to the parameter settings in the existing work [20]. The size of the input data and tasks required for the number of CPU cycles are respectively [200, 400] kB and $\left[1 \times 10^9, 5 \times 10^9\right]$, each server's CPU cycles is in [1, 2] GHz random selection. The bandwidth is set up as 20 MHZ, transmission energy p_i^{tra} is set to 36 dBm, receiving noise power N_0 is set to 2×10^9, and the channel power gain $p_{i,j}^{gain}$ is set as $-40d^{-4}$, where d is the distance between each server, energy consumption of per second p_j^{com} set as [5, 20] w, w is short for watt.

Figure 3(a) shows that the total delay of the greedy assignment method is reduced by 40.39% compared with the non-assignment method. Compared with the greedy assignment method, the delay of the random assignment method is reduced by 80.14%. This is not surprising, because our method is to divide the task into multiple subtasks and assign them to multiple servers for processing in a distributed way. Compared with the random assignment method, the delay of the MTMS method is reduced by 40.93%. This is because MTMS is to assign subtasks based on the minimum energy consumption on the server. In response, our time will be less than random assignment. From the figure, we can see that as the number of tasks increases, the total delay of tasks will be reduced more and more by using the optimization method. Therefore, our proposed method is suitable for large-scale tasks assignment scenarios. When the number of neighbor servers is fixed, the delay of the task execution will be affected by the increased size of the input data of the task. As shown in Figure 3(b), the size of the input data increases from 400kB to 800kB and the number of tasks required for the number of CPU cycles increases accordingly. Compared with the non-assignment method, the total delay of the greedy assignment method is reduced by 41.12%. Compared to the greedy method, the random method can reduce the delay by 83.26%. Furthermore, compared with the random



(a) Total time over different number of tasks



. (b) Total time over different size of tasks

FIGURE 3. The delay of different schemes.

assignment method, the delay of the optimization method is reduced by 30.21%. This means that our approach is very suitable for handling large tasks and can be adapted to the upcoming 5G era.

As can be seen from Figure 4, the average delay of greedy assignment is reduced by 39.87% compared to the non-assignment method. Compared to the greedy assignment method, the average delay of random assignment is greatly reduced. Moreover, the average time of optimization assignment is 40.91% less than that of random assignment. From Figure 4, we can see that the time efficiency of random assignment method and optimal assignment method is much higher than that of non-assignment method and greedy assignment method, which proves that the idea of dividing tasks into multiple sub-tasks can effectively reduce the time consumption. And the MTMS algorithm has good time efficiency.

When we fixed the task size and the CPU cycles required to complete the task, as the number of server increases, our total latency will gradually decrease. As can be seen from

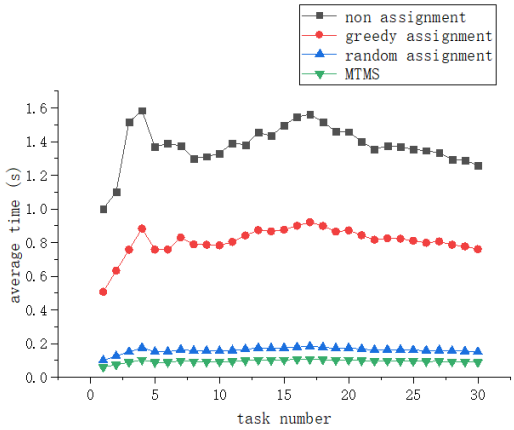


FIGURE 4. Average time over different number of tasks.

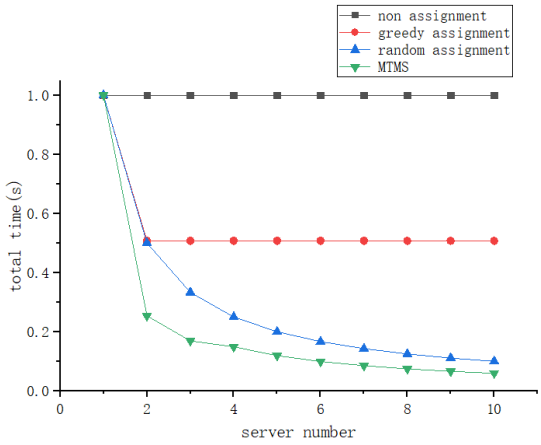
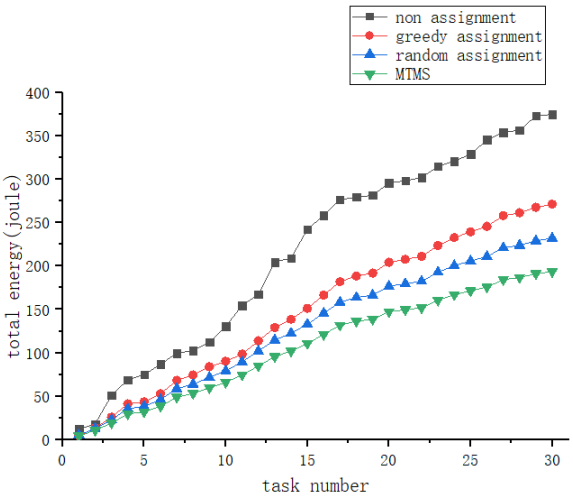


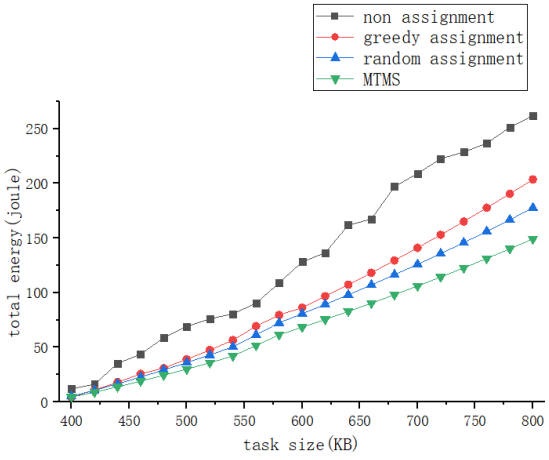
FIGURE 5. Total time of different number of servers.

Figure 5, the delay of the greedy assignment method is reduced by 44.32% compared to the non-assignment method, and the delay of the random assignment method is greatly reduced compared to the greedy assignment method. Compared to the random assignment method, the delay of the MTMS method is reduced by 29.13%. When there is only one server, it means that only the local server is processing the task, so the delay in the four schemes are the same. However, as the number of neighbor servers increases, the number of subtasks after the task division phase increases, and the data size of each subtask decreases. So the more the number of neighbor servers, the better the time efficiency of our MTMS method.

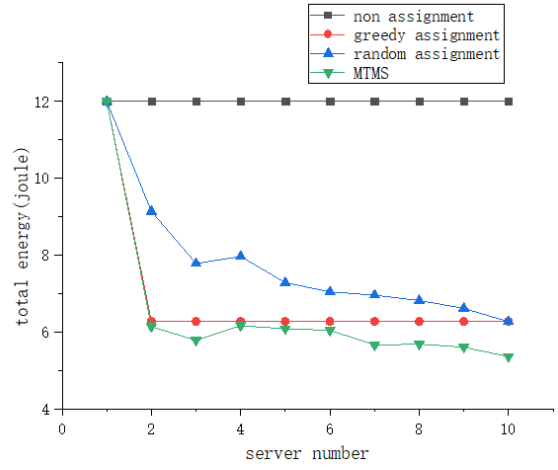
It can be seen from Figure 6(a) that the energy consumption generated by our MTMS method is reduced by 50.15% compared with the non-assignment method, and the energy consumption generated by the MTMS method is reduced by 27.74% compared with the greedy assignment method. Compared with the random assignment method, the energy consumption of the MTMS method is reduced by 16.69%. This shows that the MTMS algorithm can effectively reduce energy consumption. And the distributed task assignment method has less energy consumption than the centralized task assignment method. As shown in Figure 6(b), when the



(a) Total energy over different number of tasks.



(b) Total energy over different size of tasks.



(c) Total energy over different number of servers.

FIGURE 6. Energy consumption of schemes under different tasks and servers.

number of servers are fixed, the energy consumption of the MTMS scheme was reduced by 47.36%, 24.63%, and 15.66%, compared to the non-assignment method, the greedy assignment method, and the random assignment method respectively. As the size of the task increases, the energy

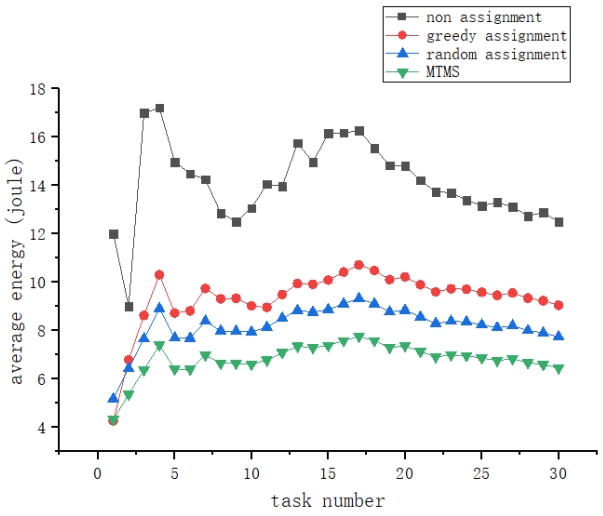


FIGURE 7. Average energy over different number of tasks.

consumption gap between the optimization algorithm and the other three distribution methods will become larger and larger, indicating that the MTMS algorithm is very suitable for processing large data volume tasks. As shown in Figure 6(c), the task input size and the required number of CPU cycles are fixed. And as the number of servers increases, the energy consumption of the MTMS method is reduced by 45.99%, 5.30%, and 16.75% compared to the non-assignment method, the greedy assignment method, and the random assignment method respectively. It can be seen from the figure that as the number of servers increase, the total energy consumption of the random assignment method will be higher than that of the greedy assignment method, so random assignment has limitations in reducing energy consumption.

It can be seen from Figure 7 that the average energy consumption generated by the MTMS scheme is reduced by 51.51% compared with the non-assignment method, and the average energy consumption generated by the MTMS method is reduced by 26.94% compared with the greedy assignment method. Compared with the random assignment method, the average energy consumption of the MTMS method is reduced by 16.70%. As can be seen from the figure, the MTMS can effectively reduce the energy consumption as the number of tasks increases, the energy consumption shows a downward trend. Therefore, the proposed MTMS is very suitable for the Internet of Things environment with a large number of devices and tasks.

VI. CONCLUSION AND FUTURE WORK

To effectively reduce the time and energy consumption of task processing, task segmentation is combined with edge server collaboration. In this work, big task can be divided and be assigned to neighbor servers to minimize the energy consumption of the whole offloading process. Therefor a Multi-subTasks-to-Multi-Severs offloading scheme (MTMS) based on the Hungarian algorithm is proposed. The MTMS algorithm can provide the optimal assignment decision for subtasks offloading, which minimizes the time and energy consumption of the entire task processing. Experiments and simulations verify the effectiveness of our ideas and methods.

In this paper, the orthogonal channel is used as a transmission channel, but in the upcoming fifth-generation (5G) era, the channels are non-orthogonal. The advantage of non-orthogonal channels is that the efficiency of the frequency can be improved, including frequency rate and bandwidth. Non-orthogonal channels can improve access quality and can transmit multiple tasks simultaneously. How to combine edge computing with 5G is the focus of our work in the future.

REFERENCES

[1] B. M. Nguyen, H. T. T. Binh, T. T. Anh, and D. B. Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment," *Appl. Sci.*, vol. 9, no. 9, pp. 1730–1749, Apr. 2019.

[2] Z. Liao, R. Zhang, S. He, D. Zeng, J. Wang, and H.-J. Kim, "Deep learning-based data storage for low latency in data center networks," *IEEE Access*, vol. 7, pp. 26411–26417, 2019.

[3] Y. T. Chen, W. H. Xu, J. W. Zuo, and Y. Kai, "The fire recognition algorithm using dynamic feature fusion and IV-SVM classifier," *Cluster Comput.*, vol. 10, pp. 1–11, Mar. 2018.

[4] X. Q. Ma, Y. Zhao, L. Zhang, H. Y. Wang, and L. M. Peng, "When mobile terminals meet the cloud: Computation offloading as the bridge," *IEEE Netw.*, vol. 27, no. 5, pp. 28–33, Sep./Oct. 2013.

[5] D. Zhang, T. Yin, G. Yang, M. Xia, L. Li, and X. Sun, "Detecting image seam carving with low scaling ratio using multi-scale spatial and spectral entropies," *J. Vis. Commun. Image Represent.*, vol. 48, pp. 281–291, Oct. 2017.

[6] M. Long, F. Peng, and Y. Zhu, "Identifying natural images and computer generated graphics based on binary similarity measures of PRNU," *Multimedia Tools Appl.*, vol. 78, no. 1, pp. 489–506, Jan. 2017.

[7] D. Cao, Y. Liu, X. Ma, J. Wang, B. Ji, C. Feng, and J. Si, "A relay-node selection on curve road in vehicular networks," *IEEE Access*, vol. 7, pp. 12714–12728, 2019.

[8] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[9] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, "The future of mobile cloud computing: Integrating cloudlets and mobile edge computing," presented at the 23rd Int. Conf. Telecommun., Thessaloniki, Greece, May 2016, pp. 1–5.

[10] D. Satria, D. Park, and M. Jo, "Recovery for overloaded mobile edge computing," *Future Gener. Comput. Syst.*, vol. 70, pp. 138–147, May 2017.

[11] M. T. Beck, M. Werner, S. Feld, and S. Thomas, "Mobile Edge Computing: A Taxonomy," presented at 6th Int. Conf. Adv. Future Internet, Lisbon, Portugal, Nov. 2014.

[12] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," presented at the 10th Int. Conf. Intell. Syst. Control, Coimbatore, India, Jan. 2016.

[13] W. Shi, H. Sun, J. Cao, Q. Zhang, and W. Liu, "Edge computing-an emerging computing model for the Internet of everything era," *J. Comput. Res. Develop.*, vol. 54, no. 5, pp. 907–924, May 2017.

[14] C. Yin, J. Xi, R. Sun, and J. Wang, "Location privacy protection based on differential privacy strategy for big data in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3628–3636, Aug. 2018.

[15] B. Xiong, K. Yang, J. Y. Zhao, and K. Q. Li, "Robust dynamic network traffic partitioning against malicious attacks," *J. Netw. Comput. Appl.*, vol. 87, pp. 20–31, Jun. 2017.

[16] S. He, W. Zeng, K. Xie, H. Yang, M. Lai, and X. Su, "PPNC: Privacy preserving scheme for random linear network coding in smart grid," *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 3, pp. 1510–1532, Mar. 2017.

[17] L. Y. Xiang, Y. Li, W. Hao, P. Yang, and X. B. Shen, "Reversible natural language watermarking using synonym substitution and arithmetic coding," *Comput. Mater. Continua*, vol. 55, no. 3, pp. 541–559, 2018.

[18] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[19] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.
[20] B. Gu, Y. P. Chen, H. J. Liao, Z. Y. Zhou, and D. Zhang, "A distributed and context-aware task assignment mechanism for collaborative mobile edge computing," *Sensors*, vol. 18, no. 8, p. 2423, Jul. 2018.
[21] M. M. Shurman and M. K. Aljarah, "Collaborative execution of distributed mobile and IoT applications running at the edge," presented at the Int. Conf. Elect. Comput. Technol. Appl., Nov. 2017.
[22] H. Wu, W. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 7, pp. 1464–1480, Jul. 2019.
[23] Y. He, J. Ren, G. Yu, and Y. Cai, "D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Trans. Commun.*, vol. 18, no. 3, pp. 1750–1763, Feb. 2019.
[24] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2015.
[25] F. Wei, S. Chen, and W. Zou, "A greedy algorithm for task offloading in mobile edge computing system," *China Commun.*, vol. 15, no. 11, pp. 149–157, Nov. 2018.
[26] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for d2d-enabled mobile-edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4193–4207, Jun. 2019.
[27] K. Zhang, Y. M. Mao, S. P. Leng, A. Vinel, and Y. Zhang, "Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks," presented at the 8th Int. Workshop Resilient Netw. Design Modeling, Halmstad, Sweden, Sep. 2016.
[28] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-aware resource allocation for edge computing," presented at the 1st Int. Conf. Edge Comput., Honolulu, HI, USA, Jun. 2017.
[29] Y. Kim, H.-W. Lee, and S. Chong, "Mobile computation offloading for application throughput fairness and energy efficiency," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 3–19, Jan. 2019.
[30] Y. Chen, N. Zhang, Y. C. Zhang, X. Chen, W. Wu, and X. S. Shen, "Energy efficient dynamic offloading in mobile edge computing for Internet of Things," *IEEE Trans. Cloud Comput.*, to be published.
[31] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4642–4655, Oct. 2018.
[32] Y. X. Jiang, M. L. Ma, M. Bennis, F. C. Zheng, and X. H. You, "A novel caching policy with content popularity prediction and user preference learning in fog-RAN," presented at the 8th Int. Workshop Resilient Netw. Design Modeling, Halmstad, Sweden, Sep. 2016.
[33] P. F. Wang, C. Yao, Z. Zheng, G. Sun, and L. Song, "Joint task assignment, transmission, and computing resource allocation in multilayer mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2872–2884, Apr. 2018.
[34] B. Gu, Z. Liu, C. Zhang, K. Yamori, O. Mizuno, and Y. Tanaka, "A stackelberg game based pricing and user association for spectrum splitting macro-femto hetNets," *IEICE Trans. Commun.*, vol. E101B, no. 1, pp. 154–162, Jan. 2018.
[35] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.
[36] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5506–5519, Aug. 2018.
[37] A. Frank, "On Kuhn's Hungarian method—A tribute from Hungary," *Nav. Res. Log.*, vol. 52, no. 1, pp. 2–5, Feb. 2007.

JIN WANG received the B.S. and M.S. degrees from the Nanjing University of Posts and Telecommunications, China, in 2002 and 2005, respectively, and the Ph.D. degree from Kyung Hee University, South Korea, in 2010. He is currently a Professor with the Changsha University of Science and Technology. He has published more than 300 international journal and conference articles. His research interests mainly include wireless sensor networks, network performance analysis, and optimization. He is a member of ACM.

WENBING WU is currently pursuing the master's degree with the Changsha University of Science and Technology. His research interests include mobile edge computing, fog computing, and machine learning. He is good at Python and C and familiar with Linux OS.

ZHUOFAN LIAO (M'14) received the Ph.D. degree in computer science from Central South University, China, in 2012. From 2017 to 2018, she was a Visiting Scholar with the University of Victoria, Canada, supported by the China Scholarship Council. She is currently an Assistant Professor with the School of Computer and Communication Engineering, Changsha University of Science and Technology, China. Her research interests include wireless networks optimization, big data, and edge computing for 5G. She has published articles in leading transactions and conferences as the first author in the above areas. She has served as a Reviewer for the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS and the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.

ARUN KUMAR SANGAIAH (M'10) received the M.E. degree from Anna University, Chennai, India, in 2007, and the Ph.D. degree from the Vellore Institute of Technology, Vellore, India, in 2014, where he is currently an Associate Professor with the School of Computing Science and Engineering. He has authored or coauthored more than 250 scientific articles in high-standard Science Citation Index (SCI) journals. His research interests include software engineering, computational intelligence, wireless networks, bioinformatics, and embedded systems.

R. SIMON SHERRATT (M'97–SM'02–F'12) received the B.Eng. degree in electronic systems and control engineering from Sheffield City Polytechnic, U.K., in 1992, and the M.Sc. degree in data telecommunications and the Ph.D. degree in video signal processing from the University of Salford, in 1994 and 1996, respectively. In 1996, he has appointed as a Lecturer in electronic engineering at the University of Reading, where he is currently a Professor of consumer electronics and the Head of the wireless and computing research. He is also a Guest Professor with the Nanjing University of Information Science and Technology, China. His research topic is on signal processing in consumer electronic devices concentrating on equalization and DSP architectures, specifically for personal area networks, USB, and Wireless USB. He has served the IEEE Consumer Electronics Society as a Vice President (Conferences), in 2008 and 2009, an AdCom Member, from 2003 to 2008 and since 2010, and the Awards Chair, in 2006 and 2007. He received the IEEE Chester Sall First Place Best TRANSACTIONS ON CONSUMER ELECTRONICS Paper Award, in 2004, and the Best Paper Award in the IEEE International Symposium on Consumer Electronics, in 2006. He served as the IEEE International Conference on Consumer Electronics General Chair, in 2009, and the IEEE International Symposium on Consumer Electronics General Chair, in 2004. He has been a member of the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS Editorial Board, since 2004, and the Editor-in-Chief, since 2011.