An LSH-Based Offloading Method for IoMT Services in Integrated Cloud-Edge Environment

XIAOLONG XU, School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China, Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing, China, Engineering Research Center of Digital Forsics, Ministry of Education, Nanjing, China, and Jiangsu Collaborative Innovation Center of Atmospheric Environmt and Equipmt Technology (CICAEET), Nanjing University of Information Science and Technology
QIHE HUANG, School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China
YIWEN ZHANG, School of Computer Science and Technology, Anhui University, Hefei, Anhui, China
SHANCANG LI, Computer Science and Creative Technologies Department, University of the West of England, UK
LIANYONG QI, School of Information Science and Engineering, Qufu Normal University, China
WANCHUN DOU, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

Benefiting from the massive available data provided by Internet of multimedia things (IoMT), enormous intelligent services requiring information of various types to make decisions are emerging. Generally, the IoMT devices are equipped with limited computing power, interfering with the process of computation-intensive services. Currently, to satisfy a wide range of service requirements, the novel computing paradigms, i.e., cloud computing and edge computing, are potential to be integrated for service accommodation. Nevertheless, the private information (i.e., location, service type, etc.) in the services is prone to spilling out during service offloading in the cloud-edge computing. To avoid privacy leakage and meanwhile improve the service utility, including the service response time and energy consumption for service executions, an Locality-sensitive-hash (LSH) based offloading method, named LOM, is devised. Specifically, LSH is leveraged to encrypt the feature information for the services offloaded to the edge servers with the intention of privacy preservation. Eventually, comparative experiments are conducted to verify the effectiveness of LOM with respect to promoting service utility.

CCS Concepts: • **Security and privacy** → **Network security**; • **Computing methodologies** → *Distributed computing methodologies*; • **Networks** → *Network services.*

Authors' addresses: Xiaolong Xu, School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China, Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing, China, Engineering Research Center of Digital Forsics, Ministry of Education, Nanjing, China, Jiangsu Collaborative Innovation Center of Atmospheric Environmt and Equipmt Technology (CICAEET), Nanjing University of Information Science and Technology, njuxlxu@gmail.com; Qihe Huang, School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China, qhuang@nuist.edu.cn; Yiwen Zhang, School of Computer Science and Technology, Anhui University, Hefei, Anhui, China, zhangyiwen@ahu.edu.cn; Shancang Li, Computer Science and Creative Technologies Department, University of the West of England, UK, Shancang.Li@uwe.ac.uk; Lianyong Qi, School of Information Science and Engineering, Qufu Normal University, China, lianyongqi@gmail.com; Wanchun Dou, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China, douwc@nju.edu.cn.

XXXX-XXXX/2020/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Additional Key Words and Phrases: IoMT, Cloud-edge computing, Service offloading, Privacy preservation, LSH

ACM Reference Format:
Xiaolong Xu, Qihe Huang, Yiwen Zhang, Shancang Li, Lianyong Qi, and Wanchun Dou. 2020. An LSH-Based Offloading Method for IoMT Services in Integrated Cloud-Edge Environment . 1, 1 (May 2020), 20 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In recent years, as connected objects and devices increase at railway speed, Internet of multimedia things (IoMT) has gained sustainable development [1][2]. With IoMT, enormous intelligent services (e.g., intelligent medical diagnostic, smart farming and real-time navigation) requiring surrounding insight are springing up [3]. Such services often rely on a gigantic amount of data for service implementation. Generally, the data analysis process of the services requires a huge amount of computing power which can be hardly satisfied by the IoMT devices [4][5]. To alleviate the negative effects on the power restrictions, drawing support from external resources with more computing ability is imperative.

Currently, cloud computing has been an extremely acclaimed way to host the IoMT services on account of the on-demand resource provision [6]. Taking advantage of cloud computing, the execution time of the IoMT services could be significantly reduced [7]. Although the resources in the cloud are guided by elastic supply to avoid uneven feeding, the energy consumption for the execution and offloading processes of the services is not easy to be eliminated [8]. Moreover, the transmission delay of offloading the cloud services and transforming the execution feedbacks is unfriendly to time-critical or real-time services.

Edge computing (EC) fully utilizes computing resources in the edge, alleviating the transmission delay significantly [9][10]. Generally, EC employs the geographically distributed edge nodes (EN) to execute time-critical or real-time services. Benefiting from the low end-to-end latency, EC greatly enables the computation-intensive and latency-critical services in IoMT [11][12]. But for the computing ability of EN is relatively limited, the efficient execution of computing-intensive services is hard to achieve. To accomplish the services that require abundant computing resources, it is meaningful to offload computing-intensive services to the cloud [13]. Thus, cloud-edge computing (EC combined with cloud computing) is potential to provide computing power of different scales, accomplishing various intelligent services [14].

Normally, individuals enjoying the intelligent services are reluctant to release their service information to the ENs in the consideration of privacy preservation [15]. Given the inherent privacy disclosure risks during propagation, it is essential to guarantee the secure service offloading in the cloud-edge computing [16]. Technically, encryption algorithm is capable of reducing privacy disclosure risks by encoding the private information. However, encrypting private information in the service means relying solely on the limited information to schedule proper service offloading, conflicting the promotion of service utility. In addition to protecting the private information, promoting service utility is at an even higher priority in the cloud-edge computing [17].

The service utility is evaluated by the service response time and the energy consumption. On the one hand, the less service response time brings about the better quality of service (QoS). On the other hand, reducing the energy consumption of executing services is a crucial criteria in the large-scale distributed computing system. But the trade-offs between decreasing offloading latency and reducing the energy expenditure are challenging to be achieved in the cloud-edge computing.

In this paper, a collaborative service offloading over big data based on Locality-sensitive-hash (LSH), called LOM, is proposed to promote utility of services and meanwhile achieve privacy preservation. The main contributions of our paper are as follows:

- The $M/M/S/\infty/\infty$ queuing model is constructed for offloading IoMT services according to the queuing theory, and then the average waiting time in the EN could be calculated.
- A cloud-edge computing paradigm is introduced to allocate external resources for data processing while the transmission delay and the execution consumption during service offloading are modeled.
- A LSH-based scheduling algorithm is leveraged to extract suitable services from massive data for collaborative offloading with privacy preservation.
- Comparative experiments are performed to demonstrate the validity and effectiveness of LOM.

The remainder of this paper is organized as follows. In Section 2, the related work of this paper is introduced. In Section 3, we present the system model and the optimization problem are presented. In Section 4, we propose the offloading method LOM which is based on LSH. Section 5 covers the experiment results and analysis. Finally, we conclude our work in Section 6.

2 RELATED WORK

IoMT is providing a solid support for the real-time multimedia services and bringing these intelligent services more sustainable development [18]. Benefiting from the involvement of multiple multimedia wireless devices and sensors, IoMT is capable of propagating more more types of data contents, i.e., audios, images, videos, etc [19]. For the IoMT devices are powerless to execute the computation-intensive and real-time services, cloud-edge computing is leveraged to provide IoMT with tremendous computing resources.

Cloud-edge computing combined with terminals has obtained increasing advancement due to its effect on extending computing capability for intelligent services. Ren et al. [20] focused on the collaboration between the cloud and the edge, and they formulated a joint allocation problem in terms of computation and communication resources to optimize the propagation latency. Lin et al. [21] contributed to the data placement of scientific workflow, and they optimized the transmission latency among different data centers by the combination of cloud and edge computing. Hao et al. [22] studied how to assign the tasks into cloud and edge while proposed a two-layer collaborative paradigm to address the allocation problem. Ruan et al. [23] presented a three-tier cloud-edge framework to improve executing performance and minimize the service latency. Thai et al. [24] proposed a generic framework of edge-cloud computing with the intention of offering horizontal and vertical offloading among service terminals. Hong et al. [25] studied the computation offloading in the edge-cloud environment and adopted a game theory based method to promote the service quality.

To efficiently decide where to offload enormous services in the integrated edge-cloud computing, investigating a proper method is imperative. Moreover, to avoid privacy disclosure, the offloading method is supposed to be integrated with privacy protection technique [26]. He et al. [27] developed a privacy-aware task offloading method using the generic Lyapunov optimization framework with the premise of minimizing cost for mobile edge computing. Xu et al. [28] put forward a joint offloading method to obtain the balance between the privacy preservation and computation utility. Min et al. [29] protected the usage pattern and the user location of the healthcare devices by changing the rate of offloading. Bai et al. [30] conceived an energy-aware offloading technique with a stress on the information security.

LSH is a potential technique to encrypt the sensitive information meanwhile retrieve the suitable services within satisfactory time. Hu et al. [31] constructed the LSH index tables based on the effective features which are extracted by the neural network to detect the repeated image. Shao et al. [32] proposed an innovative fly LSH algorithm for the wireless multimedia sensor networks,

and solved the problem of conducting rapid query over the high-dimensional and large-scale data. Ding et al. [33] developed the two-step LSH which generates the binary codes and maintains the semantic similarity for big data retrieval system. Wu et al. [34] analyzed the mechanism of consistent weighted sampling and proposed an improved algorithm to estimate the similarity with more accuracy. Li et al. [35] focused on the collision produced by the LSH-based similarity, and they proposed a network-efficient method to reduce the network cost of distributed processing framework. Guo et al. [36] built a privacy-aware index structure by LSH to equip the IoT applications with more efficiency and security.

However, most researchers neglect the application of LSH to protecting individual privacy in the cloud-edge computing. Apart from the privacy preservation, the service response time and the energy consumption are supposed to improve the efficiency of services. In view of this, an LSH-based offloading method for IoMT services in cloud-edge is designed.

3 SYSTEM MODEL AND PROBLEM DEFINITION

In this section, an IoMT service offloading framework is constructed in the integrated cloud-edge environment. Additionally, the communication model, energy consumption model and service delay model are respectively built and expounded. The key terms and corresponding descriptions are listed in Table 1.

Table 1. Key Terms and Descriptions

Term	Descriptions
J	The quantity of ENs
N	The quantity of multimedia terminals
S	The quantity of VMs in an EN
ED	The set of ENs, $ED = \{ed_1, ed_2, \cdots, ed_J\}$
MT	The set of multimedia terminals, $MT = \{mt_1, mt_2, \cdots, mt_N\}$
F_j^e	The computing rate of the j-th EN
f^{cd}	The computing rate in the cloud data center
η^{cd}	The operating power of one VM in the j-th EN
η_j^e	The operating power in the cloud data center
w_i	The data size of the i-th service
m_i	The computation size of the i-th service
r_{i,y_i}	The propagation rate between the i-th terminal and the y_i-th EN

3.1 An IoMT Service Offloading Framework in Cloud-edge Computing

As depicted in Fig. 1, an IoMT service offloading framework which is composed of N terminals providing services, J ENs and a cloud data center is constructed. In addition to an edge server (ES), an EN is provided with a base station (BS) to support offloaded services by its enough computing resources. Each EN is assumed to associate with densely distributed IoMT devices through wireless signal. As to some computation-intensive services, EN could offload them to the cloud data center which is equipped with abundant computing resources via backbone network. For the backbone

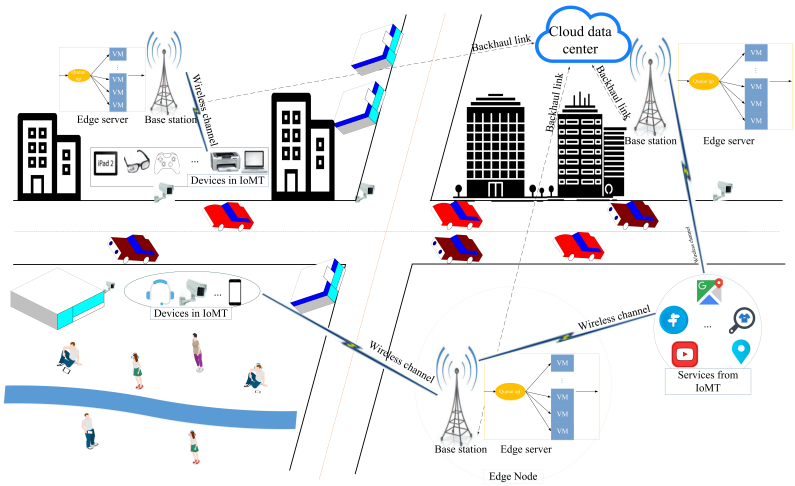

Fig. 1. An IoMT service offloading framework in edge-cloud environment

network is obsessed with network congestion, the speed of data transmission is considered to be slow [11].

Devices set is represented by $MT = \{mt_1, mt_2, \cdots, mt_N\}$, each device equipped with only one service requesting for offloading due to its limited capabilities. Each service is considered to arrive simultaneously and the computing part of each service is assumed to be atomic and is unable to split [12]. As for a terminal mt_i, $i \in MT$, the offloading service of it is donated by $T_i = (w_i, m_i)$, where m_i denotes the CPU cycles accomplishing the service requires and w_i represents the service data size.

Let $ED = \{ed_1, ed_2, \cdots, ed_J\}$ indicates the collection of ENs. Denote ed_j ($ed_j \in ED$) as the j-th EN. Then the j-th EN computation ability is defined as F_j^e. In addition, each EN is provided with S virtual machines (VM) which averagely share the computing resources of EN. Thus, the computing power of one VM at j-th EN is represented as

$$f_j^e = \frac{F_j^e}{S}. \tag{1}$$

In our model, one VM is capable of executing only one service. Besides, service could be delegated to the EN within server hop counts by optical fiber. Furthermore, to overcome the long waiting time in the edge, EN could propagate the service to the other ENs or cloud data center when it is overloaded.

3.2 $M/M/S/\infty/\infty$ Queuing Model of Services in EN

As mentioned above, each EN is considered to have S VMs and its set is represented by $VI = \{Z_1, Z_2, \cdots, Z_S\}$. As for an EN, we consider its arriving offloaded services obeys Poisson distribution with parameter of μ. Analogously, the service time is assumed to follow Poisson distribution and its parameter is expressed by λ. Then $M/M/S/\infty/\infty$ queuing model for offloaded services is applied in EN. According to our queuing model, when the system enters stable sate, the service intensity of one VM could be denoted by

$$\rho = \frac{\mu}{\lambda}, \tag{2}$$

And the service intensity of one EN is

$$\rho_S = \frac{\mu}{S \cdot \lambda}. \tag{3}$$

Define P_k as the possibility that there equals k services offloaded to the EN. Then the probability that the size of the EN being idle could be calculated, and it is denoted as

$$P_0 = \frac{1}{\sum_{n=0}^{S-1} \frac{1}{n!} \cdot \rho^n + \frac{\rho_S}{S!(1-\rho_S)}}. \tag{4}$$

Based on P_0, the possibility that there k services in the EN waiting execution is calculated , and it is expressed as

$$P_k = \begin{cases} \frac{(\mu/\lambda)^k}{k!} \cdot P_0, & (1 \le k \le S), \\ \frac{(\mu/\lambda)^k}{S!S^{(k-S)}} \cdot P_0, & (k > S). \end{cases} \tag{5}$$

When there are not enough free VMs in the EN, the offloaded services are supposed to line up until there are free VMs in the EN. Derived from (3) and (4), mean queue length of services could be calculated by taking limit and doing accumulation and it is expressed as

$$\begin{aligned} L &= \sum_{k=S+1}^{\infty} (k-S) \cdot P_k = \frac{P_0 \cdot \rho^S}{S!} \sum_{k=S+1}^{\infty} (k-S) \cdot \rho_S^{k-S} \\ &= \frac{P_0 \cdot \rho^S}{S!} \cdot \frac{d}{d\rho_S} (\sum_{k=1}^{\infty} k \cdot \rho_S^k) = \frac{P_0 \cdot \rho^S \cdot \rho_S}{S!(1-\rho_S)^2} (k \ge S). \end{aligned} \tag{6}$$

Based on (6), the average waiting time t^q in the EN could be dudeced as

$$t^q = \frac{L}{\mu} = \frac{P_0 \cdot \rho^S}{S!(1-\rho_S)} \cdot \frac{1}{S \cdot \lambda - \mu}. \tag{7}$$

3.3 Communicating Model From Terminals to ENs

Each terminal is assumed to be connected with the closest EN, and the range of spectrum from termial to EN is denoted by B which is a constant in our paper. In addition, the wireless channels of different ENs are considered to be orthogonal to avoid interference between ENs. Let $\zeta_{i,j}$ to indicate the associating relationship between the j-th EN and the i-th terminal.

$$\zeta_{i,j} = \begin{cases} 0, & \text{no connection} \\ 1, & \text{otherwise.} \end{cases} \tag{8}$$

For one terminal is associated with one EN, the $\zeta_{i,j}$ is assumed to subject to

$$\sum_{i}^{N} \sum_{j}^{J} \zeta_{i,j} = 1. \tag{9}$$

Define y_i-th EN as the EN directly connected to the i-th terminal through wireless channel. Then let M_{i,y_i} as the channel gain that differs slightly for different wireless channel, and the propagation power is denoted as p_{i,y_i}. Thus, based on Shannon formula, the propagation rate between i-th terminal and y_i-th EN could be given by

$$r_{i,y_i} = B \cdot \log_2(1 + \frac{p_{i,y_i} \cdot M_{i,y_i}}{p_0}), \tag{10}$$

where p_0 is Gaussian white noise power that is assumed to be a constant during the whole offloading. (10) implies that the high transmission power of the terminal could reduce the transmission delay but at the expense of energy expenditure. Given the quite small output size compared with service input size, the transmission from ENs to terminals is neglected.

3.4 Energy And Delay Model

In what following, the delay and energy model will be presented in terms of the execution in the EN, the execution in the cloud data center and the data transmission. Additionally, the number of terminals is considered to be unchanged during service offloading.

Generally, terminal sends offloading request for its service and then it will be received by its associated EN. Then the EN decides where the service is supposed to be offloaded, the cloud data center or the EN within serval hop counts. A decision variable $\sigma_i \in \{0, 1\}$ is defined to reveal the offloading destination for the i-th service, where $\sigma_i = 1$ expresses the service is received by an EN at the edge while $\sigma_i = 0$ shows that service is propagated to the cloud data center.

3.4.1 The Analysis of Wireless Transmission. In our model, we assume each terminal propagates its service to the connected EN directly with no local computing. The latency of the i-th terminal propagating service to its connected EN could be expressed as

$$t_i^{mt} = \frac{w_i}{r_{i,y_i}}. \tag{11}$$

The energy consumption during wireless communication between the i-th terminal and its connected EN is given by

$$e_i^{mt} = t_i^t \cdot p_{i,y_i}. \tag{12}$$

3.4.2 The Analysis of Services at Edge Layer. As to σ_i=1, the offloaded service is received by edge layer and it will be propagated to a EN within hop counts of its connected EN. The hop counts between ENs is expressed by $h_{y_i,j}$ which is defined by Manhattan distance. Denote θ as the propagation delay of one hop. The communication latency between the j-th EN and y_i-th EN is calculated as

$$t_{y_i,j}^{te} = (\theta + \frac{w_i}{\chi}) \cdot h_{y_i,j}. \tag{13}$$

where χ is the optical fiber transmission speed. When the service is offloaded to its connected for execution, the communication latency between ENs could be nonexistent.

Then, the energy expenditure and computing latency could be respectively expressed as

$$e_{i,j}^{ee} = w_i \cdot \eta_j^e, \tag{14}$$

$$t_{i,j}^{ee} = \frac{m_i}{f_j^e} + t_j^q, \tag{15}$$

where η_j^e is the operating power of one VM in the j-th EN and t_j^q is queuing latency.

3.4.3 The Analysis of Services in Cloud Layer. As to σ_i=0, the offloaded service is received by cloud layer and it will be propagated to the cloud data center to achieve more powerful computing ability. The latency of propagating the i-th service to the cloud data center is presented by

$$t_i^{cd} = \psi + \frac{m_i}{f^{cd}} + \frac{w_i}{v}, \tag{16}$$

where ψ is congestion latency on the backbone network, f^{cd} is the computing ability of the cloud data center and v is the normal propagation rate of backbone network.

The operating power of cloud is denoted as η^{cd}. Thus, energy expenditure of accomplishing the i-th service in the cloud be given by

$$e_i^{cd} = w_i \cdot \eta^{cd}. \tag{17}$$

3.5 Problem Definition

As mentioned above, there are numerous services in IoMT applying for offloading on account of poor computing resources in the terminals. To improve the QoS in IoMT, minimizing service response time and energy expenditure is extremely important. For the service offloading in the cloud-edge environment has been analyzed in the above, the energy consumption and service response time could be given by

$$E_i = e_i^{mt} + \sigma_i \cdot e_{i,j}^{ee} + (1 - \sigma_i) \cdot e_i^{cd}. \tag{18}$$

$$T_i = t_i^{mt} + \sigma_i \cdot (t^q + t_{y_i,j}^{te} + t_{i,j}^{ee}) + (1 - \sigma_i) \cdot t_i^{cd}, \tag{19}$$

We devote to minimizing energy consumption and response time of all the services. Thus, our optimization target is formulated as

$$P_1 \min \sum_{i=1}^{N} (t_i^{mt} + \sigma_i \cdot (t^q + t_{y_i,j}^{te} + t_{i,j}^{ee}) + (1 - \sigma_i) \cdot t_i^{cd}), \tag{20}$$

$$s.t. \sum_{j=1}^{s} f_j^e \le F_j^e, f_j^e \ge 0, \tag{21}$$

$$\sum_{i=1}^{N} (1 - \sigma_i) \cdot f^{cd} \le F^{cd}, f^{cd} \ge 0, \tag{22}$$

$$f_j^e \le f^{cd}, \forall j \in [1, J], \tag{23}$$

$$P_2 \min \sum_{i=1}^{N} (e_i^{mt} + \sigma_i \cdot e_{i,j}^{ee} + (1 - \sigma_i) \cdot e_i^{cd}), \tag{24}$$

$$s.t. (21) - (23),$$

where (21) and (22) imply that the allocated computing resources for services are not supposed to surpass the maximal available resources supplied by the EN and cloud, and (23) means the computing resource allocated by the EN to each service cannot exceed the computing resources allocated by cloud.

4 LOM DESIGN

In this section, the design of LOM to offload IoMT services with privacy preservation is expounded and presented at length. Firstly, the hash index table is established using LSH. Then each EN conducts service retrieval to receive IoMT services. In the last, the overview of LOM is explicated.

4.1 Hash Index Table Established With LSH

Since there are massive services in the IoMT, retrieving suitable services for offloading is supposed to be time-consuming. By establishing the hash index tables using LSH, EN is able to retrieve services for offloading efficiently. In addition, the data security could be achieved, for hashed passwords are irreversible and thus can't be decrypted. LSH is based on LSH Cluster to reduce the dimensionality of complex data, and the key elements in the LSH Cluster is the variables obeying p-stable distribution.

Definition 1 (p-stable distribution) For any d real numbers, i.e., $\beta_1, \beta_2, ..., \beta_d$, we can randomly choose d independent identically distributed variables, i.e., $\alpha_1, \alpha_2, ..., \alpha_d$. If these variables are sampled from p-distributions, the random variable $\sum_{i=1}^{X} \beta_i \cdot \alpha_i$ will have the same distribution with the variable $(\sum_{i=1}^{X} |\beta_i|^p)^{1/p} \cdot \tau$, where τ is a random variable tallying with p-distribution. It is

well-known that the p-stable distribution exists for any $p \in (0, 2]$. In this paper, we choose $p = 1$, in which case the p-stable distribution will be standard normal distribution.

Definition 2 (LSH Cluster) The LSH cluster Y ($Y = \{y : A^d \rightarrow R\}$) is a cluster of functions mapping data from d dimensions in domain A to the set of real numbers. The cluster Y is (r_1, r_2, p_1, p_2)-sensitive when it meets

$$P[y_m(v) = y_m(q)] > p_1, \ (\|v - q\| < r_1), \tag{25}$$

and

$$P[y_m(v) = y_m(q)] < p_1, \ (\|v - q\| > r_1), \tag{26}$$

for any $v, q \in A^d, y_m \in Y$, where P indicates the possibility vector v and vector q are put into the same hash bucket. The probability that vectors hashed into the same value is in proportion to their previous distance. Thus, the cluster Y is called locality-sensitive.

In LOM, the feature vector of offloading service are hashed by LSH and thus the services requiring for similar computing resources are probable to gain same hash value. Firstly, the maximal number of hop counts in ENs is determined, which is denoted by Lim_{hop}. Then, feature vectors of each service connected to its agreeable EN are constructed. In this paper, the dimension of each feature vector for service is d, composed of time delay, energy consumption etc. We set $v_{i,j} = \{\beta_{i,j,1}, \beta_{i,j,2}, ..., \beta_{i,j,d}\}$ as the j-th service feature vector belonging to the i-th EN. Afterwards, the hash process will be done, and the LSH function that corresponds to Gaussian distribution is proposed by

$$y_{a,c}(v_{i,j}) = \left\lfloor \frac{a \cdot v_{i,j} + c}{o} \right\rfloor = \left\lfloor \frac{\sum_{n=1}^{d} \alpha_n \cdot \beta_{i,j,n} + c}{o} \right\rfloor, \tag{27}$$

where α_n ($n \leq d$) is a variable independently and randomly sampled from the standard normal distribution, $o \in Z^+$ is a positive integer called quantization step and c is a positive integer chosen from (0,o]. Each LSH function $y_{a,c}$ will map $v_{i,j}$ from d-dimension vector onto an integer. The similar vector has high possibility to be mapped onto proximal integer or the same integer. Generally, the local sensitivity could be proven from a geometrical point. The projection $a \cdot v_{i,j}$ maps each feature vector $v_{i,j}$ onto a real line. As for two vectors $v_{i,j}$ and $v_{i,k}$, the distance between their hash value, $\left|a \cdot v_{i,j} - a \cdot v_{i,k}\right|$, follows the same distribution with $\left\|a \cdot v_{i,j} - a \cdot v_{i,k}\right\|_p \cdot \tau$, where τ is a variable randomly chosen from the Gaussian distribution. Furthermore, the real line is cut into segments based on the quantization step o and then the feature vectors of services will be distributed to different segment according to its hash code, which can preserve the local sensitivity of each vector.

To reduce the probability of mapping dissimilar service to the same real number and improve the discriminating power of hash process, a second function cluster G ($G = \{g : A^d \rightarrow U^k\}$) is constructed. The second function $g(v) = (y_1(v), y_2(v), ..., y_k(v))$ is constructed by picking up k hash functions $y(v)$ in the LSH Cluster $Y\{y : A^d \rightarrow R\}$. Relying on the function $g(v)$, the feature vector $v_{i,j}$ of the j-th service could be mapped from d dimensions in domain A to k dimensions in domain U. The hashed vector can be expressed as $Ut = \{u_1, u_2, ..., u_k\}$.

In addition, to increase the stability of choosing hash functions and hold the high query accuracy, L functions are randomly select from the cluster $G = \{g : A^d \rightarrow U^k\}$ to conduct multiple groups of hash process. Then these feature vectors of services will be saved in hash buckets of L hash tables. When conducting vector query service, ENs just go through all L hash tables and take the certain quantity of services as the final results.

Note that store all possible hash buckets would waste a lot of space, LSH applies other two functions F and F' to hash the bucket,

$$F(u_1, u_2, ..., u_k) = ((\sum_{i=1}^{k} r_i \cdot m_i) \bmod prime) \bmod N, \tag{28}$$

and

$$F'(u_1, u_2, ..., u_k) = (\sum_{i=1}^{k} r_i' \cdot m_i) \bmod prime, \tag{29}$$

where N is the number of feature vectors for services, prime is a big prime number $2^{32} - 3$, and r_i and $r_i^{'}$ are random real numbers. The function F and F' contribute to constructing the index based on the hash bucket of each vector. for all L hash tables, and the amount of space complexity can be decreased from O(k) to O(1).

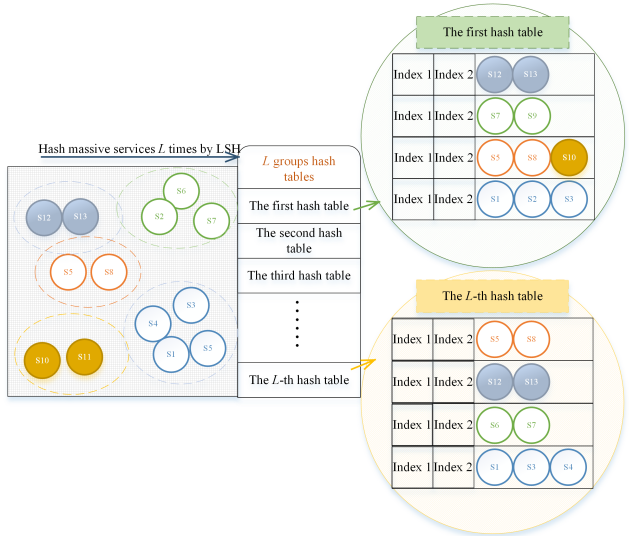

Fig. 2. An example of constructing hash tables from service 1 to service 13 by LSH.

Algorithm 1 Constructing hash table by LSH

Input: $w_i, m_i, f_j^e, \eta_j^e, B, Lim_{hop}, L$
Output: Hash index table

```
Generate LSH Cluster and randomly choose L groups, each group including k hash functions
for j = 1 to J do
    Pick the j-th EN from ED
    Receive its offloading requests and add them to the list Require
    Calculate the feature vector of the service list Require according to (7), (18) and (19)
    for l_num = 1 to L do
        while Require is not empty do
            Pick a service x from Require
            Calculate LSH index for x in the i-th hash table according to (27), (28) and (29) and
add it to the i-th hash table
            Delete x from Require
        end while
    end for
end for
return Hash index tables
```

As shown in Algorithm 1, each EN will construct L groups of hash tables by LSH to obtain short query time. First, LSH cluster is generated based on random number of Gaussian distribution and then choose L groups, each group including k hash functions. With the energy consumption model and delay model, the feature vector of each service could be calculated. Algorithm 1 elaborates how feature vectors of services are hashed by LSH. Each EN goes through its covering services and initializes the feature vectors of them. Then these feature vectors are hashed by LSH cluster, formulating L hash tables and calculating hash indexes.

Take Fig. 2 for example, the services requesting for offloading are usually divided into different groups, and these in the same group are supposed to own the similar feature vectors. Then, these services will be mapped into in L hash tables, and the similar services have high probability to be put into same bucket. It can be seen from Fig. 2 that service 1, service 2, service 3 and service 4 have similar feature vectors. In the hash index tables, service 1, service 2 and service 3 are mapped into same group in the first hash table while service 1, service 3 and service 4 are mapped into same group in the L-th hash table. In addition, for LSH is based on probability theory, there will be some handful errors during the identification of similarity. As it can be seen, service 5, service 8 and service 10 are put into same group in the first hash table though service 10 is not similar with service 5. But for services are offloaded in a large scale, these handful classification errors are subtle and could be neglected in the analysis.

Algorithm 2 Service retrieving

Input: ED, F, F'
Output: Service receiving list U_j

for $j = 1$ *to* J **do**
  Pick the j-th EN from ED
  Calculate the feature vector V_j of the j-th EN
  if length($Require$)<S **then**
    $U_j \longleftarrow Require$
  else
    for $i = 1$ *to* l **do**
      Initialize $r_{len} = length(Require)$
      for $r = 1$ *to* r_{len} **do**
        Pick the r-th service from $Require$
        Intializ k_1 and k_2 as the two hash index in the i-th table of the r-th service
        if $F(V_j) == k_1$ && $F'(V_j) == k_2$ **then**
          Add the r-th service to the U_j
          Delete the r-th service in $Require$
        end if
      end for
    end for
  end if
end for

The service retrieval process is interpreted in Algorithm 2. Benefiting from the inherent advantage of maintaining sensitivity of LSH, the hashed values of previous similar services have high possibility to be resemble, which are convenient for each EN to retrieve suitable services. Besides, once the EN has sufficient free resources for the services requiring for offloading, it would receive all the services to make advantages of its computing capacity.

4.2 Method Overview

Algorithm 3 Overview of LOM

Input: S, w_i, m_i,hash index table
Output: $Time, Energy$

Set $Time = 0$ and $Energy = 0$
Calculate the hash index tables according to Algorithm 1
for $j = 1$ *to* J **do**
    Initialize U_j according to Algorithm 2
    Pick the j-th EN from ED
    Set $len = length(U_j)$
    for $i = 1$ *to* len **do**
        Pick the n-th service as the i-th in U_j
        if $t_n^{te} + \frac{m_n}{f_j^e} < t_n^{cd}$ **then**
            Strike out n in U_j
            Offload n-th service to the cloud
            Update $Time$ and $Energy$
        else
            Strike out n in U_j
            Offload n-th service to the EN
            Update $Time$ and $Energy$
        end if
    end for
end for

LOM is based on LSH to retrieve suitable offloading service for ENs with quite short time and high accuracy. Each EN will construct feature vectors for all the services at first. Then the vectors are mapped into different hash value by LSH according to Algorithm 1. Afterwards each EN initialize its own feature vector based on its computing ability and operating state. Then, the EN could quickly retrieves the proximal vector of its feature vector in hash tables to gain suitable services according to Algorithm 2.

The overview of LOM is presented in Algorithm 3. Each EN will retrieve the suitable services in all hash index tables. If the number of its retrieved services is smaller than the number of its VMs, then these services will be all offloaded to the EN for next judgment. If the number of retrieved services is bigger than the number of its VMs, then these retrieved hashed vectors will be compared by the Euclidean distance to the hashed feature vector of EN and the first S (the number of VMs) will be offloaded to the EN for next judgment. Additionally, the queuing latency of ENs is compared to the propagation latency on the backbone network to the cloud data center. Thus, we could offload some long-time waiting services to the cloud while most are accomplished in the EN. In the last, a handful of remaining services will be all offloaded to the cloud for rich computing resources.

The programming chart of LOM is present on the Fig .3. Firstly, the LSH function clusters, which are based on p-stable distribution variables $a_1, a_2 ... a_d$, are generated. Then, an EN is picked from the EN collection, and the feature vector of the EN which is with the same formula of service feature vector is calculated according to its computing ability and energy consumption level. EN receives the service offloading requests meanwhile add them to the request list. Afterwards, the feature vector of services in the request list are hashed by the L groups LSH function clusters to construct L hash index tables for the corresponding EN. Practically, the two hash key values of the

Fig. 3. The programming chart of LOM.

hashed feature are calculated by (28) and (29). Then the service would be retrieved by the two hash key values by the feature vector of the ES. Moreover, the VM capacity of server and the queuing time are taken into account to give decision to the ES whether the service will be propagated to the cloud.

5 EXPERIMENTAL EVALUATION

5.1 Experiment Setup

The simulation parameters setting for evaluating LOM are set in Table 2.

Table 2. Simulation Parameters

Parameter	Value	Parameter	Value
N	[1000,4500]	B	40 MHZ
J	1000	f_j^e	[4,8] Gigacycles/s
S	5	η_j^e	[0.4,1] J/Gigacycle
w_i	[2,5] M bits	f^{cd}	15 Gigacycles/s
m_i	[2000,5000] Megacycles	η^{cd}	0.8 J/Gigacycle
p_0	-120 dBm	Lim_{hop}	5

In this section, abundant results are shown to examine the above analysis and validate the behavior of our method. The covering radius for each EN is 500m in the emulation. Multiple terminals served by the corresponding EN via the radio channels. are stochastically distributed in the covering area of ENs. According to [11], the channel gain M_{i,y_i} is subject to exponential distribution with parameter of 1 and the noise power density p_0 is -120 dBm. We consider that the size of each service is [2,5] Mbits and each service requires [2000,5000] Megacycles for execution. Generally, there are ten VMs in each EN to bear the offloaded services and the computing ability of each VM is set by [4,8] gigacycles/s and the energy consumption is [0.4,1] J/Gigacycle. To avoid excessive offloading distance in the edge layer, we limit the hops counts range from 1 to 5. Also, the computing ability of cloud equally allocated to each service is 15 Gigacycles/s and the energy expenditure is 0.8 J/Gigacycle. The service scale ranges from 1000 to 4500 and the growth is 500. The number of VMs in each EN is set as 5. The propagation rate on the optimal cable and the backhaul link is set as 2.997×10^8 m/s. The step length o is set as 4 according to [34], which could promote the precision rate of service retrieval level.

5.2 Comparative Algorithms

Time-greedy (TG), energy-greedy (EG) and all-to-cloud (ATC) are three comparative algorithms leveraged to conduct offloading in this paper. In the experiment, the total number of services is set from 1000 to 4500 and the interval that is a constant is 500. Then the performance of our method will be compared by the above comparative algorithms with the number of services, and the introductions of three comparative algorithms are as follows.

- TG: Services requiring for offloading are sorted based on the running time for each EN. Then the EN will receive offloaded services until its VMs are all not free. In the last, the cloud data center will receive remaining services not yet offloaded.
- EG: Compared to TG, EG chooses to sort services requiring for offloading based on the energy consumption for each EN. Similarly, each EN is supposed to receive offloaded services as long as it has free VMs and the other services will be offloaded to the cloud.

- ATC: ENs drop out in ATC algorithm and there is only one cloud data center to receive offloaded services. Although ATC is supposed to gain less execution time, the congestions on the backhauls could not be neglected.

5.3 Analysis on Energy Consumption

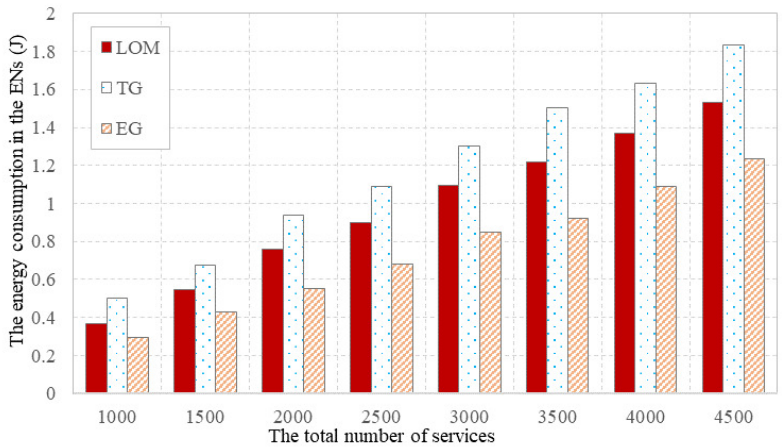

Fig. 4. Energy consumption in the ENs compared by LOM, TG and EG.

5.3.1 Analysis on Energy Consumption in The ENs . As it can be seen in Fig. 4, EG consumes the least energy in the ENs and TG maintains the most consumption of energy. LOM keeps mild energy expenditure compared to TG and EG. Since EG owns great stability on reducing energy consumption, it always keeps lowest executing energy consumption. LOM consumes slightly higher executing energy in the ENs in the majority of service scales, for it takes other energy consumption factors (e.g. propagation expenditure) together. It reveals that LOM focuses on reducing the overall energy consumption not limited to some parts.

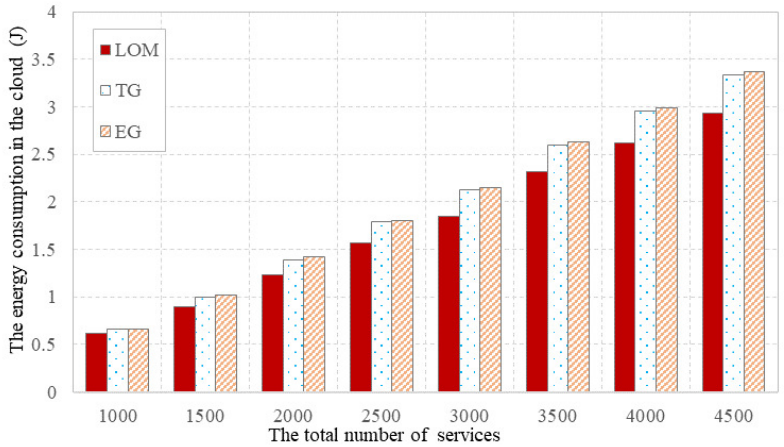

Fig. 5. Energy consumption in the cloud compared by LOM, TG and EG.

5.3.2 Analysis on Energy Consumption in The Cloud. Fig. 5 compares energy consumption in cloud by TG, EG and LOM with different number of services. EG consumes less energy than TG but more energy than LOM. LOM produces the least energy consumption among TG, EG and LOM, for it is capable of offloading services to the cloud. It is evident that LOM could better conduct overall service scheduling.

5.3.3 Analysis on Total Energy Consumption. The comparison of total energy consumption is showed in Fig. 6. We add the ATC strategy to the comparison where all the services will be offloaded to the cloud for parallel cloud computing. From Fig. 6 it can be seen that ATC produces the most energy consumption because it expends a huge amount of energy on the backhauls during high uploading period. The energy expenditure of TG is in the nick of ATC for it doesn't lay stress on the energy consumption level when making offloading decisions. In line with our expectations, LOM consumes similar energy with EG, maintaining relatively low energy consumption with the number of services throughout the whole simulation. Furthermore, LOM produces less energy than EG with increasing services, which implies stability of LOM could be improved with the increasing number of offloaded services.

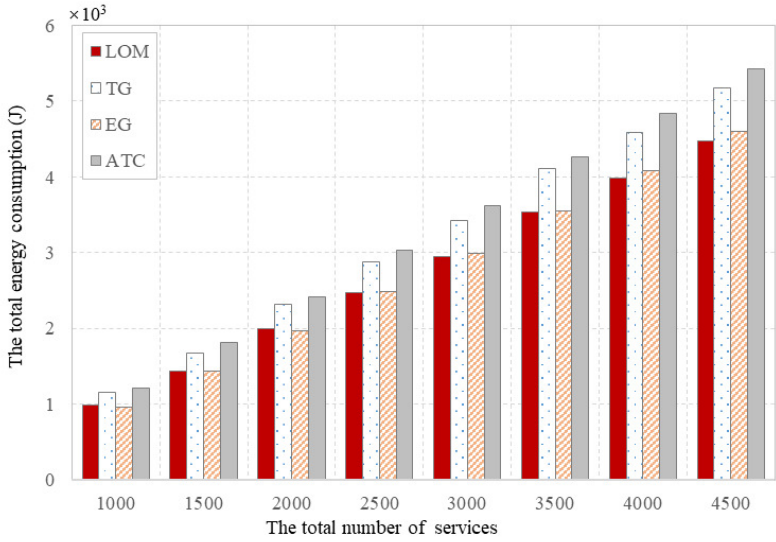

Fig. 6. Total energy consumption with the increasing number of services compared by LOM, TG, EG and ATC.

5.4 Analysis on Service Response Time

5.4.1 Analysis on Delay in the ENs . The offloading delay in ENs is analyzed in this part. Based on the data shown in Fig. 7, the offloading delay in the ENs tends to increase stablely with the growing number of services. When the services are offloaded to the ENs , LOM expends less offloading delay than EG but still longer than TG. LOM not only focuses on decreasing the time latency but also contributes to reducing the energy consumption. On this ground, LOM couldn't achieve the less delay than TG in the ENs.

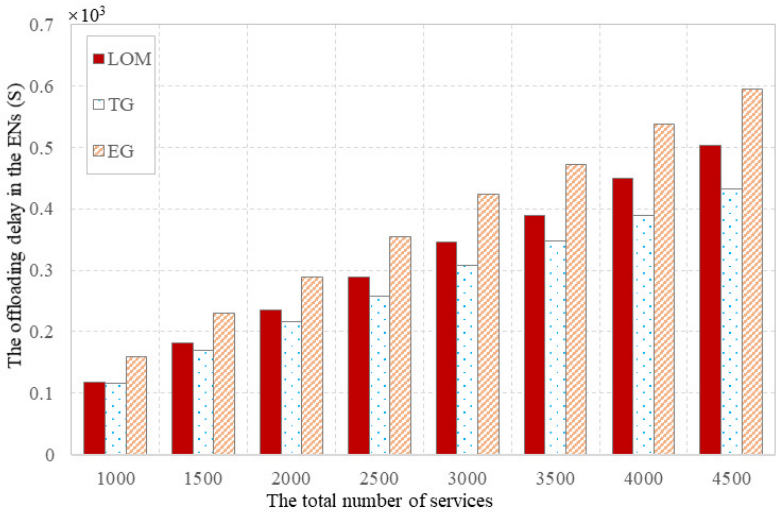

Fig. 7. Offloading delay in the ENs compared by LOM, TG and EG.

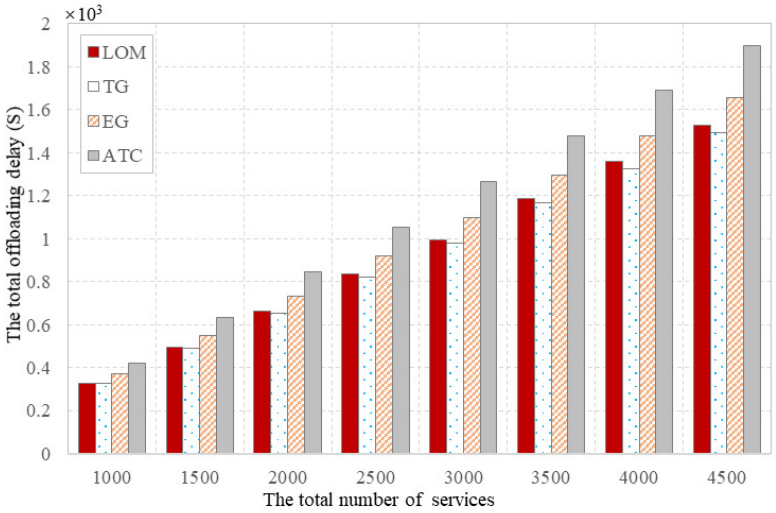

Fig. 8. Total offloading delay with the number of services compared by LOM, TG, EG and ATC.

5.4.2 Analysis on Total Delay. As expanded in Section II, the propagation time of services offloaded to the associated EN, the waiting time, the propagation time of services offloaded from the associated EN to the final server and the executing time of services make up the total offloading delay. The comparison of the total offloading delay is presented in Fig. 8. It can be seen that ATC needs the most offloading time, for the increasing services may lead to the congestion on the backhauls, which will produce serious transmission delay. LOM consumes less offloading latency than EG and ATC, for our method gives higher priority to service response time. In addition, LOM keeps close delay with TG, and the difference of the total offloading delay between all scales of services is subtle.

6 CONCLUSION AND FUTURE WORK

To acquire short IoMT service response time, low energy consumption and better preserving individual privacy in edge-cloud environment , LOM is designed. Firstly, the private information is encrypted by hashing the feature vector of each service with LSH technique. Then the hash index tables will be constructed, which are convenient for each EN to retrieve suitable services and receive offloading requests. Afterwards, the remaining services requiring for more computing resources is propagated to cloud. In our next work, we will distribute to transferring LOM to the real scene, taking account of unique features in IoMT.

ACKNOWLEDGMENT

This research is supported by the Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps under grant no. 2020DB05, and the National Natural Science Foundation of China under grant no.61702277, no.61872219 and no. 61672276. This research is also supported by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund.

REFERENCES

[1] S. He and W. Wang. Multimedia upstreaming cournot game in non-orthogonal multiple access internet of things. *IEEE Transactions on Network Science and Engineering*, 7(1):398–408, 2020.

[2] Z. Zhang, R. Sun, X. Wang, and C. Zhao. A situational analytic method for user behavior pattern in multimedia social networks. *IEEE Transactions on Big Data*, 5(4):520–528, 2019.
[3] X. Zhou, W. Liang, K. I. Wang, H. Wang, L. T. Yang, and Q. Jin. Deep learning enhanced human activity recognition for internet of healthcare things. *IEEE Internet of Things Journal*, pages 1–1, 2020.
[4] C. Singhal, C. F. Chiasserini, and C. E. Casetti. Emb: Efficient multimedia broadcast in multi-tier mobile networks. *IEEE Transactions on Vehicular Technology*, 68(11):11186–11199, 2019.
[5] X. Zhou, W. Liang, K. I. Wang, and S. Shimizu. Multi-modality behavioral influence analysis for personalized recommendations in health social media environment. *IEEE Transactions on Computational Social Systems*, 6(5):888–897, 2019.
[6] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and Y. Xiang. Block design-based key agreement for group data sharing in cloud computing. *IEEE Transactions on Dependable and Secure Computing*, 16(6):996–1010, 2019.
[7] G. Skourletopoulos, C. X. Mavromoustakis, G. Mastorakis, J. M. Batalla, H. Song, J. N. Sahalos, and E. Pallis. Elasticity debt analytics exploitation for green mobile cloud computing: An equilibrium model. *IEEE Transactions on Green Communications and Networking*, 3(1):122–131, 2019.
[8] J. Son and R. Buyya. Priority-aware vm allocation and network bandwidth provisioning in software-defined networking (sdn)-enabled clouds. *IEEE Transactions on Sustainable Computing*, 4(1):17–28, 2019.
[9] L. Wang, L. Jiao, J. Li, J. Gedeon, and M. Mühlhäuser. Moera: Mobility-agnostic online resource allocation for edge computing. *IEEE Transactions on Mobile Computing*, 18(8):1843–1856, 2019.
[10] X. Zhou, W. Liang, K. I. Wang, R. Huang, and Q. Jin. Academic influence aware and multidimensional network analysis for research collaboration navigation based on scholarly big data. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2018.
[11] Z. Tian, W. Shi, Y. Wang, C. Zhu, X. Du, S. Su, Y. Sun, and N. Guizani. Real-time lateral movement detection based on evidence reasoning network for edge computing environment. *IEEE Transactions on Industrial Informatics*, 15(7):4285–4294, 2019.
[12] E. Li, L. Zeng, Z. Zhou, and X. Chen. Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, 19(1):447–457, 2020.
[13] Y. Zhu, Q. He, J. Liu, B. Li, and Y. Hu. When crowd meets big video data: Cloud-edge collaborative transcoding for personal livecast. *IEEE Transactions on Network Science and Engineering*, 7(1):42–53, 2020.
[14] X. Hu, L. Wang, K. Wong, M. Tao, Y. Zhang, and Z. Zheng. Edge and central cloud computing: A perfect pairing for high energy efficiency and low-latency. *IEEE Transactions on Wireless Communications*, 19(2):1070–1083, 2020.
[15] M. Jia, Z. Yin, D. Li, Q. Guo, and X. Gu. Toward improved offloading efficiency of data transmission in the iot-cloud by leveraging secure truncating ofdm. *IEEE Internet of Things Journal*, 6(3):4252–4261, 2019.
[16] Q. He, B. Li, F. Chen, J. Grundy, X. Xia, and Y. Yang. Diversified third-party library prediction for mobile app development. *IEEE Transactions on Software Engineering*, pages 1–1, 2020.
[17] S. Han, X. Xu, S. Fang, Y. Sun, Y. Cao, X. Tao, and P. Zhang. Energy efficient secure computation offloading in noma-based mmtc networks for iot. *IEEE Internet of Things Journal*, 6(3):5674–5690, 2019.
[18] M. A. Jan, M. Usman, X. He, and A. Ur Rehman. Sams: A seamless and authorized multimedia streaming framework for wmsn-based iomt. *IEEE Internet of Things Journal*, 6(2):1576–1583, 2019.
[19] K. Thiyagarajan, R. Lu, K. El-Sankary, and H. Zhu. Energy-aware encryption for securing video transmission in internet of multimedia things. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(3):610–624, 2019.
[20] J. Ren, G. Yu, Y. He, and G. Y. Li. Collaborative cloud and edge computing for latency minimization. *IEEE Transactions on Vehicular Technology*, 68(5):5031–5044, 2019.
[21] B. Lin, F. Zhu, J. Zhang, J. Chen, X. Chen, N. N. Xiong, and J. Lloret Mauri. A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing. *IEEE Transactions on Industrial Informatics*, 15(7):4254–4265, 2019.
[22] F. Hao, D. Park, J. Kang, and G. Min. 2l-mc3: A two-layer multi-community-cloud/cloudlet social collaborative paradigm for mobile edge computing. *IEEE Internet of Things Journal*, 6(3):4764–4773, 2019.
[23] L. Ruan, Y. Yan, S. Guo, F. Wen, and X. Qiu. Priority-based residential energy management with collaborative edge and cloud computing. *IEEE Transactions on Industrial Informatics*, 16(3):1848–1857, 2020.
[24] M. Thai, Y. Lin, Y. Lai, and H. Chien. Workload and capacity optimization for cloud-edge computing systems with vertical and horizontal offloading. *IEEE Transactions on Network and Service Management*, 17(1):227–238, 2020.
[25] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng. Multi-hop cooperative computation offloading for industrial iot–edge–cloud computing environments. *IEEE Transactions on Parallel and Distributed Systems*, 30(12):2759–2774, 2019.
[26] X. He, R. Jin, and H. Dai. Deep pds-learning for privacy-aware offloading in mec-enabled iot. *IEEE Internet of Things Journal*, 6(3):4547–4555, 2019.

[27] X. He, R. Jin, and H. Dai. Peace: Privacy-preserving and cost-efficient task offloading for mobile-edge computing. *IEEE Transactions on Wireless Communications*, 19(3):1814–1824, 2020.

[28] X. Xu, C. He, Z. Xu, L. Qi, S. Wan, and M. Z. A. Bhuiyan. Joint optimization of offloading utility and privacy for edge computing enabled iot. *IEEE Internet of Things Journal*, 7(4):2622–2629, 2020.

[29] M. Min, X. Wan, L. Xiao, Y. Chen, M. Xia, D. Wu, and H. Dai. Learning-based privacy-aware offloading for healthcare iot with energy harvesting. *IEEE Internet of Things Journal*, 6(3):4307–4316, 2019.

[30] T. Bai, J. Wang, Y. Ren, and L. Hanzo. Energy-efficient computation offloading for secure uav-edge-computing systems. *IEEE Transactions on Vehicular Technology*, 68(6):6074–6087, 2019.

[31] W. Hu, Y. Fan, J. Xing, L. Sun, Z. Cai, and S. Maybank. Deep constrained siamese hash coding network and load-balanced locality-sensitive hashing for near duplicate image detection. *IEEE Transactions on Image Processing*, 27(9):4452–4464, 2018.

[32] W. Shao, R. Xiao, J. Huang, H. Liu, and X. Du. Fjlt-flsh: More efficient fly locality-sensitive hashing algorithm via fjlt for wmsn iot search. *IEEE Internet of Things Journal*, 6(4):7122–7136, 2019.

[33] K. Ding, C. Huo, B. Fan, S. Xiang, and C. Pan. In defense of locality-sensitive hashing. *IEEE Transactions on Neural Networks and Learning Systems*, 29(1):87–103, 2018.

[34] W. Wu, B. Li, L. Chen, C. Zhang, and P. S. Yu. Improved consistent weighted sampling revisited. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2332–2345, 2019.

[35] H. Li, S. Nutanong, H. Xu, c. YU, and F. Ha. C2net: A network-efficient approach to collision counting lsh similarity join. *IEEE Transactions on Knowledge and Data Engineering*, 31(3):423–436, 2019.

[36] C. Guo, J. Jia, Y. Jie, C. Z. Liu, and K. R. Choo. Enabling secure cross-modal retrieval over encrypted heterogeneous iot databases with collective matrix factorization. *IEEE Internet of Things Journal*, 7(4):3104–3113, 2020.