Resource Cooperative Scheduling Optimization Considering Security in Edge Mobile Networks

Cheng Fang[1,3], Peng Yang[1,2(✉)], Meng Yi[1,2], Miao Du[1,2], and Bing Li[1,2]

[1] Key Laboratory of Computer Network and Information Integration, Southeast University, Ministry of Education, Nanjing, China
pengyang@seu.edu.cn
[2] School of Computer Science and Engineering, Southeast University, Nanjing, China
[3] School of Cyber Science and Engineering, Southeast University, Nanjing, China

Abstract. With the rapid development of technologies such as the Internet of Things and artificial intelligence, the contradiction between limited user computing resources and real-time, fast, and safe processing of large amounts of data has become an urgent issue. The emergence of edge computing provides IoT applications with a low-latency, high-bandwidth, and high-performance computing service. Due to the complexity and dynamics of the edge computing environment itself, and the limited resources of the terminal, the security issue of resource collaborative scheduling in the edge mobile network has become an important research topic. Different from existing work, this paper proposes an efficient and secure multi-user resource cooperative scheduling model, which comprehensively considers resource allocation, task offloading, QoE requirements, and data security. In the model, ChaCha20 encryption technology is introduced as a security mechanism to prevent data from being maliciously stolen by attackers during the offloading process, and computing speed is used as an indicator to quantify QoE requirements. A resource collaborative scheduling algorithm that integrates security mechanisms and computing acceleration is also proposed to minimize the total cost of optimizing the edge computing system. Finally, the effectiveness and superiority of the model and algorithm are verified by simulation experiments.

Keywords: Edge Computing · Edge Mobile Network · Resource Collaborative Scheduling · Security Mechanism · Computing Acceleration

1 Introduction

In recent years, with the support of mobile Internet, especially 5G technology, the Internet of Things (IoT) has developed rapidly. Various applications of production and life, i.e., self-driving cars, drone flight, Virtual Reality/Augmented Reality (VR/AR), and mobile healthcare are emerging. The novel scenarios require not only very low and deterministic network latency, but also massive, heterogeneous, and diverse data access. The centralized computing and processing model of traditional cloud computing faces tremendous computational and network pressure and cannot meet the demands of the

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 561, pp. 54–74, 2024.
https://doi.org/10.1007/978-3-031-54521-4_4

Internet of Everything [1]. In this context, the edge computing system becomes an effective solution [2]. In edge computing, to better support high-density, high-bandwidth, and low-latency service scenarios, an effective way is to build a service platform (edge data center) on the network edge close to users, providing storage, computation, and network resources, as well as sinking some critical services to the edge of the access network so as to reduce the loss of bandwidth and latency caused by network transmission and multi-level forwarding [3]. Figure 1 illustrates the edge computing paradigm.

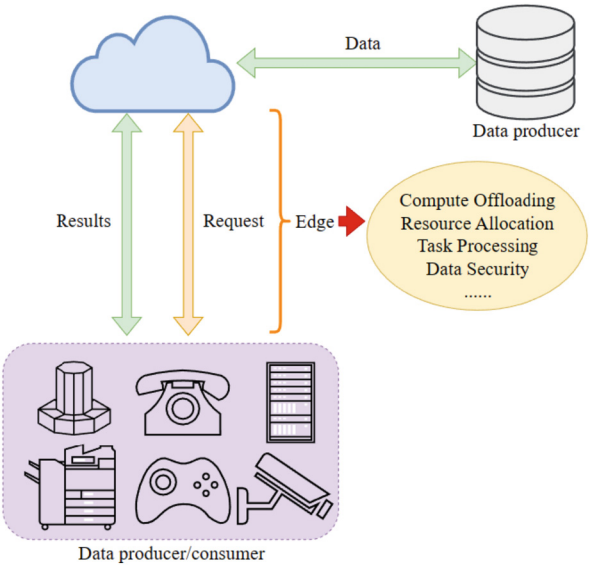

Fig. 1. Edge computing paradigm.

In edge computing, static end devices (e.g., sensors in smart homes, cameras in public places), and dynamic end devices (e.g., drones and vehicles) are involved in this environment, which give a challenge for resource management [4]. Suppose resource scheduling does not efficiently utilize dispersed resources. In that case, it can lead to under-utilization of the actual available resources, which in turn leads to as high latency and high energy consumption. In contrast, if an efficient resource scheduling strategy is applied, resources can be efficiently combined to create available and cost-effective pools of computational resources, thus optimizing the resource management problem [5]. Considering the importance of resource scheduling, several studies seek to optimize the weighted sum of latency and energy consumption in edge computing systems and try to find the trade-off replacement relationship between them. However, in practical scenarios, in addition to latency and energy consumption, the resource scheduling process focuses on some factors such as user quality of experience (QoE) and data security. Currently, more and more users start to pay attention to their subjective feelings, and their QoE demand is sensitive, while the requirement of low latency cannot completely cover the QoE demand of users, so this motivates us to focus on QoE in the process

of resource scheduling. In addition, QoE requirements are closely related to data security, as attackers can easily insert interference factors into unencrypted task data so as to achieve a good user experience, and wireless interference between mobile users can also inadvertently reduce user QoE metrics due to the shared wireless channels while encrypting task data can ensure data integrity. Therefore, it is necessary to consider the element of data security along with QoE. Currently, although some works [6, 7] have focused on the four factors of latency, energy consumption, QoE, and data security, most of the studies have jointly analyzed two or three of these factors and they have not paid sufficient attention to the tight coupling between QoE and data security, and the studies have not well combined these four factors. The trade-off among the four factors, i.e., latency, energy consumption, QoE requirements, and data security needs to be addressed.

Therefore, motivated by the above facts, this paper proposes a resource collaborative scheduling model that comprehensively considers resource allocation, computing offload, data security, and QoE for edge mobile networks. In addition, we design a resource scheduling algorithm that combines security mechanisms and computational acceleration to minimize the cost of the entire system. The main contributions of this study are summarized as follows:

- To solve the problem of cooperative scheduling of multi-user resources in edge mobile networks considering security, we propose an efficient and secure integrated model for joint optimization of resource allocation, computation offloading, data security, and QoE requirements.
- In order to minimize the total cost of the edge computing system, we propose a resource collaborative scheduling algorithm that integrates security mechanisms and computing acceleration. By introducing ChaCha20 encryption technology as a security mechanism, data security during task offloading is guaranteed, and use computing speed as a quantified indicator for QoE requirements.
- By comparing the algorithm proposed in this paper with the benchmark algorithm, the proposed algorithm is evaluated from multiple perspectives such as energy consumption, latency, security decision, task offload rate, and QoE. Simulation results prove the validity and superiority of the proposed algorithm.

The rest of this paper is organized as follows, with related work given in Sect. 2. Section 3 describes the system integration model of this paper in terms of resource allocation, computation offloading, data security, and QoE. The problem formulation is given in Sect. 4 for the problem to be solved. In Sect. 5, the algorithm design is given in this paper. In Sect. 6, simulation experiments are performed to demonstrate the effectiveness of the proposed model and algorithms. Finally, Sect. 7 concludes the whole paper.

2 Related Work

In this section, we first present some related work with different objectives and point out their shortcomings. Then, we give the motivation for the research in this paper.

2.1 Latency or Energy Consumption

Guo *et al.* [8] formulated a task offloading and computational resource scheduling problem as an energy consumption minimization problem. They used DVFS [9] technique to directly reduce energy consumption via adjusting the processor frequency. Also, to achieve the goal of low-energy offloading, they proposed a distributed dynamic offloading and computational resource scheduling algorithm. However, they only considered the energy consumption reduction by tuning the processor frequency without considering the QoE requirements of user. Deng *et al.* [10] study the offloading problem in the framework of green and sustainable mobile edge computing in IoT systems. To minimize the overall system response time, they used the Lyapunov technique to decompose the formulation problem into a convex optimization problem, then proposed a dynamic parallel computational offloading and energy management algorithm (DPCOEM). Finally, the near-optimal solution of the algorithm is also implemented [11]. However, the solution complexity of convex optimization problems is often high, and the practicality in realistic scenarios is poor.

2.2 Latency or Energy Consumption

Lu *et al.* [12] started their analysis from a simple case and extended it to the complex cases afterward, where they modelled the multi-user resource allocation problem in edge computing and use an approximation algorithm for local search to solve the NP, achieving the goal of minimizing the cost. However, the approximation algorithm they used easily falls into a local optimum and cannot guarantee the performance of the solution. Li *et al.* [13] proposed a game-theoretic scheme to optimize the offloading strategy of joint computing-intensive and bandwidth-intensive resources to minimize the system cost. However, the optimal solution obtained from their proposed scheme may not be the global optimal solution. Meng *et al.* [14] proposed a fault-tolerant dynamic resource scheduling approach using a game-theoretic scheduling mechanism that jointly considers cost and system latency to solve the joint cost-latency problem for mobile edge computing and thus meet the requirement of minimizing the application cost by a user-defined deadline. However, their study of scheduling only under a two-tier architecture instead of a three-tier architecture is less practical. In addition, they ignore the security privacy issues during offloading. The fault tolerance of their scheduling method is greatly reduced and the security risk is elevated when there is a security attack at the edge layer.

2.3 QoE

Ning *et al.* [15] studied the intelligent scheduling problem of in-vehicle edge computing, divided the original complex problem into two relatively simple subproblems, and proposed a two-sided matching scheme and a DQN method to schedule vehicle requests meeting the strict QoE requirements of mobile smart vehicles. However, both of the above studies require huge parameters and a long learning time, which makes it difficult to balance the delay, energy consumption, and QoE requirements. In addition, both studies are a black-box process, which poses significant security risks [16]. Guo and Zhang *et al.* [17] studied the energy-efficient computation offloading management schemes for

mobile edge computing systems with small cell networks, combining the advantages of both genetic algorithms and particle swarm optimization algorithms is used to design a suboptimal algorithm to solve the computation offloading problem, that is minimizing the energy consumption of all user devices with consideration of QoE.

2.4 Data Security

Slađana Jošilo and György Dán [18] studied the secure offloading decision problem of coordinated wireless devices, they prove the existence of pure policy Nash equilibrium by building a game theoretic model of the problem and propose a polynomial complexity algorithm for computational equilibrium to solve the problem. Ibrahim *et al.* [19] studied the computational offloading and resource allocation problem of mobile edge computing and proposed a multi-user offloading model with data security, using AES [20] encryption as the security layer, and integrated security, computation offloading, and resource allocation to determine the optimal computation offloading decision for all mobile users to minimize the time and energy consumption of the whole system. Zhang *et al.* [21] studied the ARM-based big. LITTLE processor architecture in edge environment [22, 23] for the energy-efficient task offloading problem, they presented the task offloading problem as a computational resource scheduling problem and proposed a computational resource scheduling algorithm (MUCRS) to minimize the time and energy costs by finding the pure policy Nash equilibrium point through a game theoretic model. In order to strengthen the security of transmitted data, Xu et al. [24] proposed a computational offloading method called BeCome that supports blockchain, and uses a non-dominated sorting genetic algorithm to generate a balanced resource allocation strategy.

2.5 Motivation

From the given analysis of the existing studies although many different types of solutions have been given for different research objectives, most of the research objectives are too low dimensioned and do not jointly consider the additional impact that QoE, for example, some users who are using latency-sensitive applications are very strict in their QoE requirements in addition to high latency requirements. Latency requirements cannot simply be equated with QoE requirements. It is also observed that while a portion of the work has studied the joint optimization of latency, energy consumption and QoE, the security of the data transmission during offloading has not been adequately considered in these research efforts. This can be fatal in some scenarios, as resource scheduling is meaningless if data security risks occur during the process of resource scheduling in edge computing systems. In addition, some work focuses on jointly optimizing the four aspects of latency, energy consumption, QoE requirements, and data security, unfortunately, the solutions provided by these studies do not weigh the above four factors well, some work fails to tightly couple QoE requirements with data security, while some work gives a compromise solution that sacrifices energy consumption for latency and QoE improvements, which is not suitable for resource-constrained devices.

In summary, focusing on these issues mentioned above, the work in this paper aims to consider resource allocation and computational offloading, and focuses on the tight

coupling between QoE and data security in order to minimize the overall cost of edge computing systems and seek a tradeoff between the four aspects of latency, energy consumption, QoE, and data security.

3 System Model

This section focuses on the system model used in this study. The architecture of the edge computing system is given in Fig. 2. Specifically, the multi-user task offloading in the mobile edge computing system, where $M = \{1, 2,\ldots, N\}$ is defined as a set of mobile users (MUs) connected to a single Macro Base Station (MBS), and the mobile edge server is located in the MBS. Each mobile user device generates various tasks. For easy reference, some of the notations used in this section are summarized in Table 1.

Usually, we can describe any task $T = \{D, c, \alpha, \gamma, \tau\}$. Thus, we can use $T_i = \{D_i, c_i, \alpha_i, \gamma_i, \tau_i\}$ to represent each MU i-generated task. Where D_i denotes the data size of T_i generated by MU i, c_i denotes the processing density (in CPU cycles/bit) of T_i, which may vary and can be obtained by the task analyzer [23, 24]. α_i $(0 \leq \alpha_i \leq 1)$ denotes the parallelizable fraction of T_i, which is an important parameter to determine the QoE. Unlike many researches that ignore the return process due to the small values of the processed results [17, 25–27], this paper pays attention to the return process of the results because it focuses on the information carried by the returned values of the results. Denote γ_i as the ratio of the data size of the returned result to the initial task data size of T_i, and τ_i as the delay constraint of T_i. Denote the wireless bandwidth of the end device assigned to task T_i as B. N MUs are in the edge computing environment, which are associated with a single macro base station and an edge server through a wireless channel. The generated task T_i can be processed locally or offloaded to the edge or cloud for processing. The offloading operation is based on different requirements such as energy consumption, latency, cost, and QoE. λ_i $(0 \leq \lambda_i \leq 1)$ is denoted as the offloading decision variable of task T_i generated by MU I, representing the ratio of the offloaded data size to the total data size of task T_i. If $\lambda_i = 0$, task T_i may be processed locally; if $\lambda_i = 1$, task T_i can be completely offloaded. Otherwise, $\lambda_i D_i$ is offloaded and $(1 - \lambda_i)$ D_i is processed locally.

3.1 Local Computing Model

Task processing in local: the number of CPU cores of different MU i is denoted as n_i, the processing power (i.e., CPU frequency in cycles/sec) of each core allocated by MU i for local computation is denoted as f_i^l, then the power consumption of each core of user MU i for local processing of data is denoted as,

$$p_i^l = \kappa_1 \left(f_i^l\right)^3. \tag{1}$$

where κ_1 is a coefficient reflecting the relationship between processing power and power consumption on the user of mobile terminal device [28].The local computation time of task T_i generated by MU i: Based on Amdahl's law [29], The local computation time can

Table 1. List of symbols in the system model.

Notation	Description
M	Collection of user devices
N	Total number of mobile user (MUs)
i	Corresponding i-th MU
T_i	Tasks generated by MU i
D_i	Size of task data generated by MU i
c_i	Processing density of T_i
α_i	Parallelizable fraction of T_i
γ_i	The ratio of the data size of the returned result to the initial task data size of T_i
τ_i	Delay constraint of T_i
λ_i	Offloading decision variables for MU i-generated task T_i
n_i	Number of CPU cores for different MU i
f_i^l	The processing power of each core allocated by MU i for local computation
r_i	Data transfer rate for offloading data from task Ti to the edge over the wireless communication link
P_i	The transmission power of the mobile user device MU i
h_i	Channel power gain
ω_0	Channel noise power density
m_i	Number of cores allocated by the edge server for processing of tasks generated by different MU i
f_i^e	Processing power per core of the edge server assigned to MU i
$r_i\prime$	Data transfer rate during result return
P_0	The transmission power of the edge server
A_i^l	Local computation speed for task T_i
A_i^e	Edge computation speed for task T_i
Si	Binary security decision for each MU i
p_i^l	Power consumption per core for user MU i for local data processing
t_i^l	Local computation time for MU i
E_i^l	Local computation power consumption for MU i
p_i^e	Power consumption per core for edge processing data
t_i^e	Edge server computation time
E_i^e	Edge energy consumption
βi	Total number of CPUs required for encryption computation tasks on MU

(*continued*)

Table 1. (*continued*)

Notation	Description
ηi	Total number of CPUs required for decryption compute tasks on the edge server
t_i^s	Total time for MU i to perform compute offload operations with security decisions
E_i^s	Total energy consumption executed by MU i during computing offload operations with security decisions
C_i^l	The total cost of MU to perform all tasks locally
C_i^s	The total cost of performing all tasks on the edge server
w_i^t	Weighted parameter for execution time of MU i decisions
w_i^e	Weighted parameters of energy consumption for MU i decisions

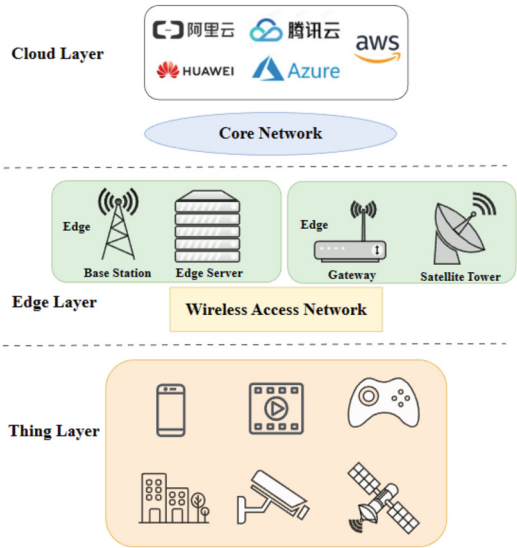

Fig. 2. Edge computing system.

be composed of the computation time of the serialized part and the computation time of the parallel part, and the computation time of the serialized part can be expressed as,

$$t_i^{ls} = \frac{c_i(1-\alpha_i)(1-\lambda_i)D_i}{f_i^l}. \tag{2}$$

The computation time of the parallel part can be expressed as,

$$t_i^{lp} = \frac{c_i\alpha_i(1-\lambda_i)D_i}{f_i^l n_i}. \tag{3}$$

From the above, it can be calculated that the local computation time of MU i can be expressed as,

$$t_i^l = t_i^{ls} + t_i^{lp} = \frac{c_i(1-\lambda_i)D_i}{f_i^l}\left(1-\alpha_i+\frac{\alpha_i}{\mathrm{n_i}}\right). \tag{4}$$

Then the locally calculated energy consumption of MU i: the locally calculated energy consumption equation is,

$$E_i^l = p_i^l t_i^{ls} + n_i p_i^l t_i^{lp} = \kappa_1 c_i D_i (1-\lambda_i)\left(f_i^l\right)^2. \tag{5}$$

3.2 Communication Model

Offloading process: data from task $\mathrm{T_i}$ is offloaded to the edge via a wireless communication link. The data transfer rate is denoted it by r_i. The data transmission rate can be characterized by a wireless transmission model based on Shannon's formula [25]. P_i is the transmission power of the mobile user device MU I, h_i is the channel power gain, ω_0 is denoted as the channel noise power density.

When data is offloaded from the mobile user device to the edge over a specified wireless bandwidth B, the data transfer rate can be expressed as,

$$r_i = B\log_2 1 + \frac{P_i h_i^2}{\omega_0 B}. \tag{6}$$

Transmission delay: Based on the above analysis, the transmission delay for offloading $\lambda_\mathrm{i}\mathrm{D_i}$ bit data to the edge can be obtained by,

$$t_i^{off} = \frac{\lambda_i \mathrm{D}_i}{r_i}. \tag{7}$$

Energy consumption: the energy consumption of the terminal device transmitting the offloaded $\lambda_\mathrm{i}\mathrm{D_i}$ bit data is expressed as,

$$\begin{aligned} E_i^{off} &= P_i t_i^{off} \\ &= \frac{\lambda_i \mathrm{D}_i P_i}{r_i} \end{aligned}. \tag{8}$$

Computation time in edge: $\lambda_\mathrm{i}\mathrm{D_i}$ bits of data are offloaded to the edge after the edge processes the data. m_i is denoted as the number of cores allocated by the edge server for processing tasks generated by different MU i. f_i^e denotes the processing power (CPU frequency in cycles/sec) of each core of the edge server allocated to MU i. In general, the processing power at edge is much larger than the processing power at the local terminal. Therefore, the following comparison equation holds,

$$f_i^e \gg f_i^l. \tag{9}$$

The power consumption of each core processing data at edge can be expressed as,

$$p_i^e = \kappa_2\left(f_i^e\right)^3. \tag{10}$$

where κ_2 is a coefficient reflecting the relationship between edge-side processing power and energy consumption The computation time of the offloaded $\lambda_i D_i$ bit data includes the computation time of the serialized part and the computation time of the parallelizable part. The computation time of the serialized part can be expressed as,

$$t_i^{es} = \frac{c_i \lambda_i (1 - \alpha_i) D_i}{f_i^e}. \tag{11}$$

The computation time of the parallel part can be expressed as,

$$t_i^{ep} = \frac{c_i \lambda_i \alpha_i D_i}{f_i^e m_i}. \tag{12}$$

From the above, it can be calculated that the edge calculation time can be expressed as,

$$t_i^e = t_i^{es} + t_i^{ep} = \frac{c_i \lambda_i D_i}{f_i^e} \left(1 - \alpha_i + \frac{\alpha_i}{m_i}\right). \tag{13}$$

Energy consumption in edge: the formula for calculating the edge energy consumption of data $\lambda_i D_i$ bits is,

$$E_i^e = p_i^e t_i^{es} + m_i p_i^e t_i^{ep} = \kappa_2 c_i \lambda_i D_i \left(f_i^e\right)^2 \tag{14}$$

Result return: After the task Ti is processed, the result can be returned to the mobile terminal device. r_i' is denoted as the data transfer rate during the result return, and P_0 is the transfer power of EN. Similar to the offloaded data transfer rate, r_i' can be expressed formally as,

$$r_i' = B \log_2 1 + \frac{P_0 h_i^2}{\omega_0 B}. \tag{15}$$

Transmission delay of result return: Based on the above analysis, the sending delay of $\gamma_i D_i$ bit result return can be expressed as,

$$t_i^{ret} = \frac{\gamma_i D_i}{r_i'}. \tag{16}$$

Energy consumption of result return: the EN energy consumption for transmission of the $\gamma_i D_i$ bits of the processing result to the mobile terminal device is expressed as,

$$E_i^{ret} = P_0 t_i^{ret} = \frac{\gamma_i D_i P_0}{r_i'}. \tag{17}$$

3.3 QoE Model

QoE requirements are a key consideration for mobile user devices before making task offloading decisions, and computation speed can be used to quantify this requirement.

Note that computation speed refers to the speed improvement ratio of processing tasks at the edge compared to local computing.

If the $(1 - \lambda_i)$ D_i bit data of task T_i is computed locally, a speedup A_i^l is obtained, and this speedup can be expressed as,

$$A_i^l = \frac{1}{(1-\alpha_i) + \frac{\alpha_i}{n_i}}. \tag{18}$$

Similarly, if the task is processed at the edge server, a speedup A_i^e is obtained, which can be expressed as,

$$A_i^e = \frac{1}{(1-\alpha_i) + \frac{\alpha_i}{m_i}}. \tag{19}$$

3.4 Data Security

Data security: In the case of computation offloading, each MU i can send data about the computation tasks to be processed to the edge server via a wireless communication channel. However, these data are prone to data security issues during transmission. Thus, security decisions need to be introduced to protect computing tasks from data security threats. ChaCha20 encryption is a new cryptography technology adopted by Google [30]. It is powerful and especially on ARM platforms with lean instruction set CPUs. ChaCha20 encryption performs three to four times better than AES in the same configuration of cell phones and it is also more adaptable to mobile environments. Compared to AES, the utilization of ChaCha20 encryption as the securing task data reduces the amount of data generated by encryption and decryption. In turn, improving the user experience, reducing waiting time, and indirectly saving battery life.

Denote $S_i \in \{0, 1\}$ as the binary security decision for each MU i, which is made individually by each MU based on the privacy requirements of the application data. $S_i = 0$ means that the mobile terminal device MU i offloads the computational task using no encryption; and $S_i = 1$ means that the mobile terminal device MU i encrypts the computational task and data using ChaCha20 encryption before transmission to the edge server. The edge server further decrypts the data after receiving the task and data, then executes the computational task and sends the processed result back to MU i.

β_i and η_i are used to denote the total number of CPU cycles required to encrypt and decrypt the data of the computational tasks at the MU and the edge server, respectively. Thus, considering the application of encryption techniques, the additional overhead in terms of time and energy for remote execution of computational tasks on the edge server can be expressed respectively,

$$t_i^{sec} = t_i^{enc} + t_i^{dec} = S_i \lambda_i \left(\frac{\beta_i}{n_i f_i^l} + \frac{\eta_i}{m_i f_i^e} \right), \tag{20}$$

$$E_i^{sec} = n_i p_i^l t_i^{enc} + m_i p_i^e t_i^{dec} = S_i \lambda_i \left(\kappa_1 \left(f_i^l \right)^2 + \kappa_2 \left(f_i^e \right)^2 \right). \tag{21}$$

Given the previous subsection, the communication process, the computational process, and the security model are considered, the total time and energy consumption of the MU i are executed when a security computational offload operation is applied,

$$t_i^s = t_i^{off} + t_i^e + t_i^{ret} + t_i^{sec}, \tag{22}$$

$$E_i^s = E_i^{off} + E_i^e + E_i^{ret} + E_i^{sec}. \tag{23}$$

Based on Eqs. (4), (5), (22), and (23), the total overhead of performing all tasks locally and remotely on the MU and edge servers in terms of execution time and energy consumption can be calculated as follows,

$$C_i^l = w_i^t t_i^l + w_i^e E_i^l, \tag{24}$$

$$C_i^s = w_i^t t_i^s + w_i^e E_i^s. \tag{25}$$

where w_i^t and $w_i^e \in [0,1]$ are denoted as the weighted parameters of execution time and energy consumption of MU i's decision, respectively. When user i is more concerned about the energy consumption of its device than the latency, $w_i^e = 1$ and $w_i^t = 0$; when MU i is executing a latency-sensitive application, the concern is more about the time than the energy consumption, so $w_i^t = 1$ and $w_i^e = 0$. We set different values of w_i^e and w_i^t for different objectives.

4 Problem Formulation

In this paper, the joint model of resource allocation, computation offloading, QoE requirements, and data security is formulated as a joint optimization problem in a system with multiple mobile users and a single-edge server. The objective of this paper is to minimize the cost of the entire system, note that cost is defined as a linear combination of latency and energy consumption, data security is defined as a security decision, and QoE requirements are quantified as computational speedup as a priori judgment condition for offloading decisions. Thus, the problem formulation is,

$$min_{\lambda_i}\left[\sum_{i=1}^{N} \lambda_i C_i^s + \sum_{i=1}^{N} (1-\lambda_i) C_i^l\right], \forall i \in N \tag{26}$$

$$\begin{aligned}
\text{s.t} \quad & \text{C1}: \left[\lambda_i E_i^s + (1-\lambda_i) E_i^l\right] \le E_i^l, \\
& \text{C2}: \textstyle\sum_{i=1}^{N} \lambda_i r_i \le B, \\
& \text{C3}: \textstyle\sum_{i=1}^{N} \lambda_i r_{i\prime} \le B, \\
& \text{C4}: \textstyle\sum_{i=1}^{N} \lambda_i m_i^{max} f_i^s \le F, \\
& \text{C5}: \min\left\{t_i^l, t_i^s\right\} \le \tau_i, \\
& \text{C6}: 0 \le \lambda_i \le 1, \\
& \text{C7}: 0 \le \alpha_i \le 1, \\
& \text{C8}: f_i^e \gg f_i^l, \\
& \text{C9}: S_i \in \{0, 1\}, \\
& \text{C10}: w_i^t \in [0, 1], \\
& \text{C11}: w_i^e \in [0, 1].
\end{aligned}$$

The objective is to minimize the cost of the whole system. The constraint C1 is to limit the total energy consumption of all MUs. Constraint C1 means that the total energy consumption of remote execution is not higher than the total energy consumption of local execution in the process of computation offloading. Constraints C2 and C3 give the bandwidth limitation of the wireless communication channel. Constraint C4 limits the upper of the processing power in edge server. Constraint C5 limits the total execution time to be less than or equal to the latency constraint, ensuring the deadline requirement for task completion. Constraint C6 ensures that the computation offloading decision ranges between 0 and 1. Constraint C7 guarantees that the parallelizable fraction is between 0 and 1. Constraint C8 indicates that the processing capacity of the edge server is larger than the processing capacity of the local endpoint. Constraint C9 indicates that the security decision is a binary offloading decision. Constraint C10 and constraint C11 indicate that the weighted parameters of execution time and energy consumption of MU i decision are in the interval from 0 to 1, respectively.

It can be observed that the problem in Eq. (26) is a mixed-integer linear programming problem where the objective function and all the associated constraints are linear. Therefore, the optimal value of the offloading decision for each MU can be obtained using the branch-and-bound method [31].

5 Algorithm Design

The algorithm designed in this paper integrates computational resources, communication resources, task offloading, user QoE requirements, and security decisions to seek a trade-off between four aspects. As shown in Algorithm 1. The algorithm gives a detailed procedure for optimal task offloading for multiple users in edge computing systems. The time complexity of the algorithm is O(N), and N denotes the total number of mobile users. The inputs to the algorithm are the task quintets of all MUs i, the corresponding kernel processing power of all MUs, and the number of additional cycles required for MU task data encryption, while the output of the algorithm is the set of optimal offloading policies corresponding to each of all MUs, and this set can be used as a policy profile.

The algorithm just starts by initializing the offloading decisions of all MUs by assigning them all a value of 0. After a given time slot t, each MU sends a reconnaissance message to the edge server and along with the number of MU self-generated cores for wireless channel transmission and fast MBS processing, the MBS returns the number of CPU cores m_i assigned to MU i. The small reconnaissance packets are ignored at the wireless channel and MBS The delay and energy consumption of the small reconnaissance packet at the wireless channel and MBS are ignored because of its tiny data volume. After obtaining the kernel information of MBS, MU computes the local computation speedup and the edge computation speedup. Only when the edge computation speedup is greater than the local computation speedup, the task offloading make sense, otherwise, the offloading decision should choose local for processing. After that, the transfer rate corresponding to all MU i offloading tasks and the transfer rate of the returned results from MBS is further calculated. Finally, the edge server is used to determine the optimal offloading decision for each MU i. Specifically, the branch-and-bound method (BAB) is used to find the set of optimal offloading decisions λ^* in Eq. (26) under the constraints C1 to C11, so as to minimize the cost of the whole system.

Algorithm 1: A Cooperative Resource Scheduling Algorithm Integrating Security Mechanism and Computing Acceleration

Input: $T_i = \{D_i, c_i, \alpha_i, \gamma_i, \tau_i\}, \forall i \in N$; Processing power f_i^l of MU; Number of additional cycles β_i

Output: The set of all MU's respective optimal offloading decisions $\lambda_i, \forall i \in N$

1. Initialization: initialize the offload decision for each MU i, i.e., $\lambda_i = 0, \forall i \in N$
2. **for** all MU i and a given time slot t **do**
3. **Transmit** $T_0 = \{1, c_i, 0, \gamma_i, \tau_i\} \cup \{n_i\}$ to MBS
4. **Return** m_i
5. **end for**
6. **while** $i \leq N$ **do**
7. $A_i^l = \frac{1}{(1-\alpha_i)+\frac{\alpha_i}{n_i}}$
8. $A_i^e = \frac{1}{(1-\alpha_i)+\frac{\alpha_i}{m_i}}$
9. **if** $A_i^l \geq A_i^e$ **then**
10. $\lambda_i = 0$
11. **else**
12. $\lambda_i > 0$ && $\lambda_i \leq 1$
13. **Calculate** $r_i = B \log_2\left(1 + \frac{P_i h_i^2}{\omega_0 B}\right), \forall i \in N$
14. $\lambda_i = BAB(min_{\lambda_i} [\sum_{i=1}^{N} \lambda_i C_i^s + \sum_{i=1}^{N} (1-\lambda_i) C_i^l], \forall i \in N)$
15. $O(\lambda_i) = \{\lambda_1, \lambda_2, \ldots, \lambda_N\}$
16. $i = i + 1$
17. **Return** $O(\lambda_i)$
18. **end for**

6 Experiment

In this section, this paper first introduces the experimental environment and parameter settings, then simulates and analyzes the proposed model and algorithm in term of energy consumption, latency, cost, security decision, task offloading rate, and QoE.

6.1 Experimental Environment and Parameter Settings

In this paper, the simulation experiments are done in the EdgeCloudSim simulation tool [32], which runs on a PC equipped with an AMD Ryzen 5 4600H CPU, 3.0 GHz frequency, and 16 GB RAM capacity, and which runs on the Windows 11 Business Editions platform. An edge computing system with 60 MUs is considered, and the number of CPU cores per MU is randomly assigned from the set {2, 4, 6, 8}, which ensures the randomness of the number of cores in different MU. The corresponding computational capacity of each MU is randomly allocated from the set {0.4, 0.5,…, 1.0} GHz, which can better distinguish the heterogeneous capacity among different MU. The CPU computing power of the edge server is set to 20 GHz. The data size of each

MU-generated task is between 2000 KB and 5000 KB, and the workload of the task ranges from 500 to 1500 (Megacycles). The transmission power of each mobile user device MU i is fixed at 100 mW. The corresponding wireless channel bandwidth is set to 20 MHz, and the corresponding channel gain and channel noise are 10^{-3} and 10^{-9}, respectively. Finally, w_i^t and w_i^e are randomly assigned from the set {0, 0.2, 0.5, 0.8, 1.0} to meet different requirements. In addition, the number of additional CPU cycles required to encrypt and decrypt the transmission data is set to 200 Megacycles. To ensure the rationality of the security decisions, the security decisions are set randomly in the simulation experiments, which means that the security decisions are made by the mobile user devices to decide whether to adopt them or not. In order to reduce the error of the experiment, 50 simulations were performed and the results were summed and averaged after 50 different results were obtained to ensure the rigor of the experiment.

6.2 Experimental Results and Analysis

Figure 3 shows the relationship between the task offloading rate of mobile user MUs and the total number of mobile users in both cases, with and without the security decision. It can be observed that when the number of mobile users MUs is between 0 and 40, the task offloading rate of users in both cases with and without security decisions is close to 100%, while when the number of mobile users exceeds 40, the task offloading rate starts to decrease in both cases. The reason is that when the number of mobile users increases, mobile users choose to offload tasks to the edge server, which inevitably causes congestion in the wireless communication channel, and the final result is that the overall task offload rate decreases for everyone.

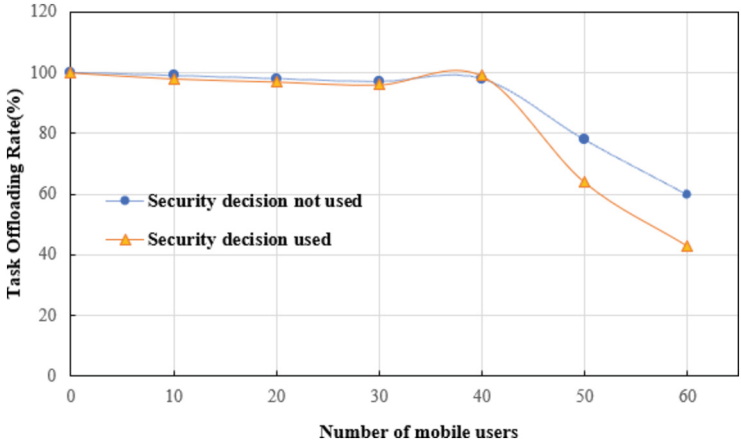

Fig. 3. Task offloading rate of mobile users.

Figure 4 shows the variation of the calculated acceleration value. As the number of users becomes larger, the mutual interference between MUs becomes more frequent due to the shared wireless communication channel. Mutual interference between MUs

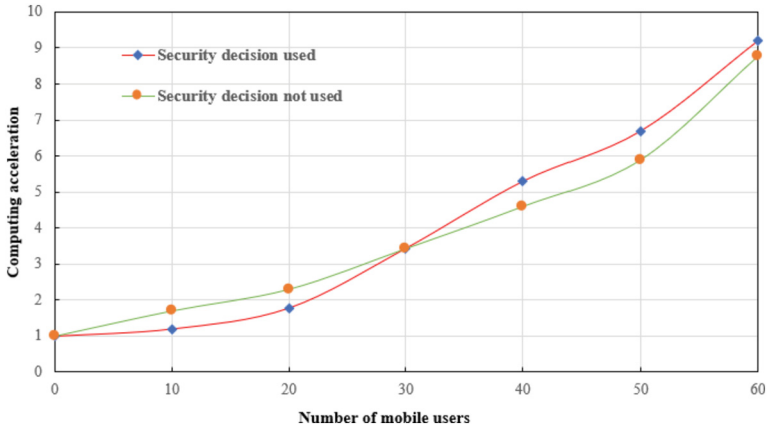

Fig. 4. Calculated acceleration values for mobile users.

becomes more frequent, and when the security decision is adopted. An encryption technique can guarantee the data integrity and reduce the mutual interference between MUs in a certain procedure, which in turn improves the value of computational acceleration.

The analysis in terms of task offloading rate and QoE have been given above, and we compares the three aspects of latency, energy consumption, and cost. Figure 5 reflects the relationship between the number of mobile users and energy consumption. From the figure, it can be seen that the proposed model is lower than the literature [33, 34] in terms of energy consumption, regardless of whether the model proposed in this paper uses security decisions or not because the model and algorithm have a good resource scheduling strategy. When the security decision is used, the energy consumption when the security decision is naturally more than the energy consumption when the security decision is not used because both MU and edge server need to operate on additional encryption and decryption, which requires additional energy consumption.

Figure 6 illustrates the relationship between the number of mobile users and the average latency. It can be noted that the curves in the literature [33, 34] become steeper and their average latency grows faster as the number of mobile subscribers increases. This is because the two aforementioned works do not sufficiently focus on the solutions of the trade-off problem, which in turn leads to increasingly steeper curves and longer delays. In contrast, the model proposed in this paper has a flatter growth rate of the curve with or without the security decision. Note that the lowest average latency is observed in the case where no security decision is used because the MU and edge server do not require additional CPU cycles to process the encryption and decryption operations in the case where no security decision.

Finally, the simulation experiments focus on the cost of the whole system, Fig. 7 shows the relationship between the number of MUs and the total cost. lt can be seen that the total cost of the proposed model in this paper is slightly higher than the literature [34] when the number of MUs is approximately less than or equal to 20. While when the number of MUs increases further, the proposed model is the lowest and the best among the three comparisons. The fact is that this paper takes into account delay, energy

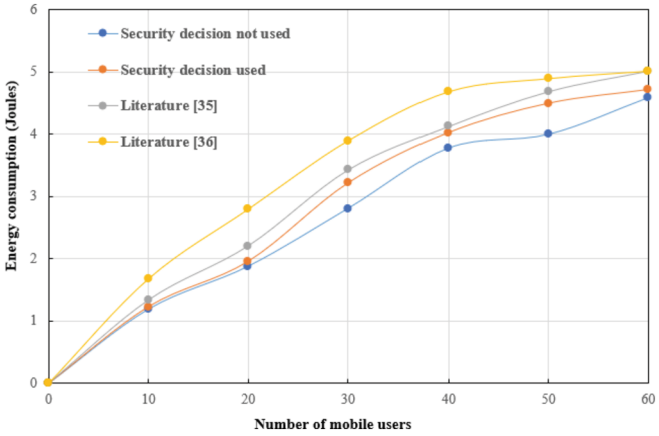

Fig. 5. Relationship between energy consumption and MU quantity.

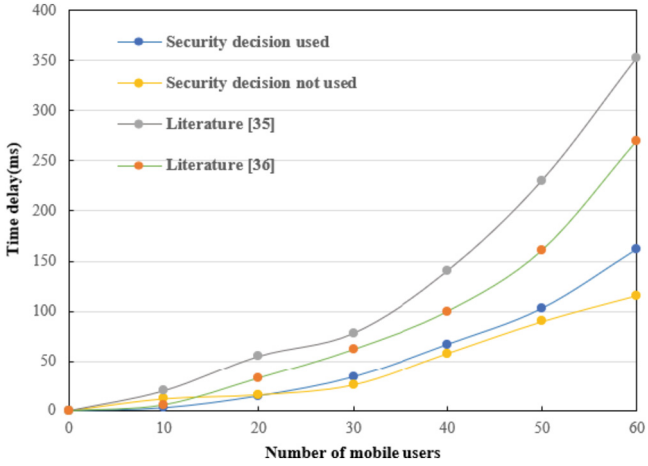

Fig. 6. Average latency versus MU quantity.

consumption, QoE, and data security and solves the trade-off problem well. Further, Fig. 8 shows the comparison between the proposed model with and without security decision and literature [33, 34]. Regarding the percentage of total cost savings, the percentage of total cost savings is the largest when the security decision is not adopted, while the percentage of total cost savings after security decision is similar to literature [33], which is due to the efficient algorithm proposed in this paper. The algorithm works well with the security decision to reduce the cost as much as possible while ensuring data security.

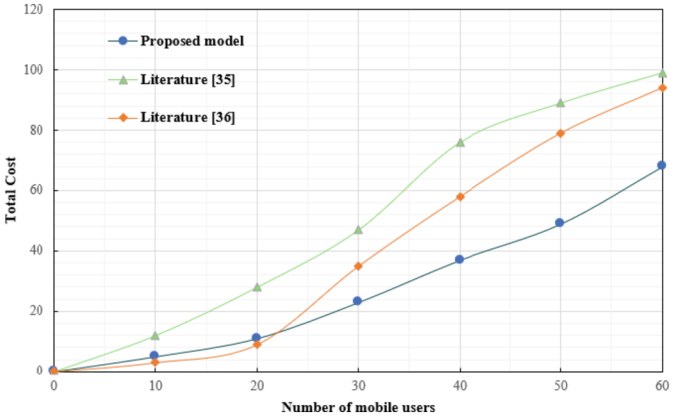

Fig. 7. Number of MU versus total cost.

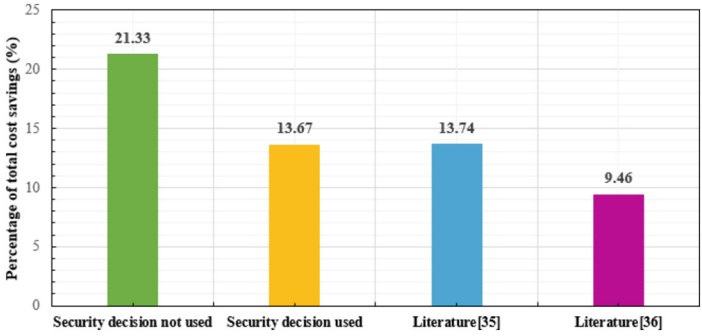

Fig. 8. Comparison of total cost savings percentage.

7 Conclusion

In this study, for edge mobile networks considering security, this paper proposes an efficient and secure multi-user resource cooperative scheduling model, which comprehensively considers resource allocation, task offloading, QoE requirements and data security, and optimizes The goal is to minimize system overhead. The problem is modeled as a mixed integer linear programming problem. To solve this problem, this paper proposes a resource cooperative scheduling algorithm that integrates security mechanisms and computing speed, and introduces ChaCha20 encryption as a security decision to prevent task data from being maliciously stolen by attackers during the offloading process. At the same time, computing speed is used as an indicator to quantify QoE requirements. In addition, this paper conducts simulation experiments and analyzes on the proposed model in terms of energy consumption, delay, cost, whether to use security decisions, task offload rate, and QoE, and compares the proposed model with benchmark values. Finally, the simulation results show that the proposed model and algorithm are effective.

Future work considers expanding the problem from a multi-user single-edge server scenario to a multi-user multi-edge server or even a multi-user multi-task multi-edge server scenario and considers a novel game-theoretic mechanism as a solution to the above problems. Another direction is to consider the problem of distributed resource scheduling (DRS) in edge computing, which is still very challenging.

Acknowledgments. This work was supported in part by the Consulting Project of Chinese Academy of Engineering under Grant 2023-XY-09, the National Natural Science Foundation of China under Grant 62272100, and in part by the Fundamental Research Funds for the Central Universities and the Academy-Locality Cooperation Project of Chinese Academy of Engineering under Grant JS2021ZT05.

References

1. Wei, S., Hui, S., Jie, C., et al.: Edge computing—an emerging computing model for the internet of everything era. J. Comput. Res. Dev. **54**(5), 907–924 (2017)
2. Jiang, C., Fan, T., Gao, H., et al.: Energy aware edge computing: a survey. Comput. Commun. **151**, 556–580 (2020)
3. Xin, H., Jun, T., Jian, L.: Collaborative trustworthy framework for edge computing. J. Electron. Inf. Technol. **44**(12), 4256–4264 (2022)
4. Luo, Q., Hu, S., Li, C., et al.: Resource scheduling in edge computing: a survey. IEEE Commun. Surv. Tutorials **23**(4), 2131–2165 (2021)
5. Xin, Z., Fang, L., Zhi, C., et al.: Edge computing: platforms, applications and challenges. J. Comput. Res. Dev. **55**(2), 327–337 (2018)
6. He, W., Zhang, Y., Huang, Y., et al.: Integrated resource allocation and task scheduling for full-duplex mobile edge computing. IEEE Trans. Veh. Technol. **71**(6), 6488–6502 (2022)
7. Lu, Y., Chen, X., Zhang, Y., et al.: Cost-efficient resources scheduling for mobile edge computing in ultra-dense networks. IEEE Trans. Netw. Serv. Manage. **19**(3), 3163–3173 (2022)
8. Guo, S., Liu, J., Yang, Y., et al.: Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing. IEEE Trans. Mob. Comput. **18**(2), 319–333 (2018)
9. Chandrakasan, A.P., Sheng, S., Brodersen, R.W.: Low-power CMOS digital design. IEICE Trans. Electron. **75**(4), 371–382 (1992)
10. Deng, Y., Chen, Z., Yao, X., et al.: Parallel offloading in green and sustainable mobile edge computing for delay-constrained IoT system. IEEE Trans. Veh. Technol. **68**(12), 12202–12214 (2019)
11. Vaidya, U., Mehta, P.G., Shanbhag, U.V.: Nonlinear stabilization via control Lyapunov measure. IEEE Trans. Autom. Control **55**(6), 1314–1328 (2010)
12. Pasteris, S., Wang, S., Herbster, M., et al.: Service placement with provable guarantees in heterogeneous edge computing systems. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 514–522. IEEE (2019)
13. Li, Q., Zhao, J., Gong, Y.: Cooperative computation offloading and resource allocation for mobile edge computing. In: 2019 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–6. IEEE (2019)
14. Meng, S., Li, Q., Wu, T., et al.: A fault-tolerant dynamic scheduling method on hierarchical mobile edge cloud computing. Comput. Intell. **35**(3), 577–598 (2019)

15. Ning, Z., Dong, P., Wang, X., et al.: Deep reinforcement learning for vehicular edge computing: an intelligent offloading system. ACM Trans. Intell. Syst. Technol. (TIST) **10**(6), 1–24 (2019)
16. Zhang, L., Zhou, W., Xia, J., et al.: DQN-based mobile edge computing for smart Internet of vehicle. EURASIP J. Adv. Sig. Process. **2022**(1), 1–16 (2022)
17. Guo, F., Zhang, H., Ji, H., et al.: An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing. IEEE/ACM Trans. Networking **26**(6), 2651–2664 (2018)
18. Jošilo, S., Dán, G.: Computation offloading scheduling for periodic tasks in mobile edge computing. IEEE/ACM Trans. Networking **28**(2), 667–680 (2020)
19. Elgendy, I.A., Zhang, W., Tian, Y.C., et al.: Resource allocation and computation offloading with data security for mobile edge computing. Futur. Gener. Comput. Syst. **100**, 531–541 (2019)
20. Daemen, J., Reijndael, R.V.: The advanced encryption standard. Dr. Dobb's J. Softw. Tools Prof. Programmer **26**(3), 137–139 (2001)
21. Zhang, J., Zheng, R., Zhao, X., et al.: A computational resources scheduling algorithm in edge cloud computing: from the energy efficiency of users' perspective. J. Supercomput., 1–22 (2022)
22. Stepanovic, S., Georgakarakos, G., Holmbacka, S., et al.: An efficient model for quantifying the interaction between structural properties of software and hardware in the ARM big. LITTLE architecture. Concurrency Comput. Pract. Exp. **32**(10), e5230 (2020)
23. Xu, X., Zhang, X., Gao, H., et al.: BeCome: blockchain-enabled computation offloading for IoT in mobile edge computing. IEEE Trans. Industr. Inf. **16**(6), 4187–4195 (2019)
24. Panneerselvam, S., Rinnegan, S.M.: Efficient resource use in heterogeneous architectures. In: Proceedings of the 2016 International Conference on Parallel Architectures and Compilation, pp. 373–386 (2016)
25. Miettinen, A.P., Nurminen, J.K.: Energy efficiency of mobile clients in cloud computing. HotCloud **10**(4–4), 19 (2010)
26. Melendez, S., McGarry, M.P.: Computation offloading decisions for reducing completion time. In: 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 160–164. IEEE (2017)
27. Wang, C., Liang, C., Yu, F.R., et al.: Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. IEEE Trans. Wireless Commun. **16**(8), 4924–4938 (2017)
28. Mao, Y., Zhang, J., Song, S.H., et al.: Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. IEEE Trans. Wireless Commun. **16**(9), 5994–6009 (2017)
29. Du, J., Zhao, L., Feng, J., et al.: Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. IEEE Trans. Commun. **66**(4), 1594–1608 (2017)
30. Amdahl, G.M.: Validity of the single processor approach to achieving large scale computing capabilities. In: Proceedings of the April 18–20, Spring Joint Computer Conference, pp. 483–485 (1967)
31. Konopiński, M.K.: Shannon diversity index: a call to replace the original Shannon's formula with unbiased estimator in the population genetics studies. PeerJ **8**, e9391 (2020)
32. Nir, Y., Langley, A.: ChaCha20 and Poly1305 for IETF Protocols (2018)
33. Boyd, S., Mattingley, J.: Branch and bound methods. Notes EE364b, Stanford University, 07 (2007, 2006)
34. Sonmez, C., Ozgovde, A., Ersoy, C.: EdgeCloudSim: an environment for performance evaluation of edge computing systems. Trans. Emerg. Telecommun. Technol. **29**(11), e3493 (2018)

35. Elgendy, I.A., Zhang, W.Z., Zeng, Y., et al.: Efficient and security multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks. IEEE Trans. Netw. Serv. Manage. **17**(4), 2410–2422 (2020)
36. Bibi, A., Majeed, M.F., Ali, S, et al.: Secured optimized resource allocation in mobile edge computing. Mob. Inf. Sys. **2022** (2022)