sensors

MDPI

Article

Energy-Aware Computation Offloading of IoT Sensors in Cloudlet-Based Mobile Edge Computing

Xiao Ma [1], Chuang Lin [1], Han Zhang [2,*] and Jianwei Liu [2]

[1] Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China; maxiao13@mails.tsinghua.edu.cn (X.M); chlin@tsinghua.edu.cn (C.L.)

[2] School of Cyber Science and Technology, Beihang University, Beijing 100191, China; liujianwei@buaa.edu.cn

* Correspondence: zganghanhan@gmail.com

Received: 24 May 2018; Accepted: 11 June 2018; Published: 15 June 2018

Abstract: Mobile edge computing is proposed as a promising computing paradigm to relieve the excessive burden of data centers and mobile networks, which is induced by the rapid growth of Internet of Things (IoT). This work introduces the cloud-assisted multi-cloudlet framework to provision scalable services in cloudlet-based mobile edge computing. Due to the constrained computation resources of cloudlets and limited communication resources of wireless access points (APs), IoT sensors with identical computation offloading decisions interact with each other. To optimize the processing delay and energy consumption of computation tasks, theoretic analysis of the computation offloading decision problem of IoT sensors is presented in this paper. In more detail, the computation offloading decision problem of IoT sensors is formulated as a computation offloading game and the condition of Nash equilibrium is derived by introducing the tool of a potential game. By exploiting the finite improvement property of the game, the Computation Offloading Decision (COD) algorithm is designed to provide decentralized computation offloading strategies for IoT sensors. Simulation results demonstrate that the COD algorithm can significantly reduce the system cost compared with the random-selection algorithm and the cloud-first algorithm. Furthermore, the COD algorithm can scale well with increasing IoT sensors.

Keywords: mobile edge computing; QoS-aware; energy-aware; Internet of Things; heterogeneous wireless access

1. Introduction

With the rapid emergence and sharp growth of Internet of Things (IoT), enormous computation and communication requirements are induced, exceeding the capacity of current data centers and mobile networks [1–4]. Mobile edge computing, e.g., cloudlet-based mobile edge computing [5] and fog computing [6,7], introduces promising solutions to overcoming this challenge in a distributed manner. In mobile edge computing, ubiquitous communication and computation resource provisioning is enabled by shifting computation resources to the edge of the Internet, i.e., wireless access networks [8–11]. Due to the advantages of low latency and high bandwidth, many efforts have been devoted to exploring the potentials of mobile edge computing in practical applications, such as IoT [12] caching [13–15], security services [16] and vehicular networks [17]. However, the resources of mobile edge computing are limited, especially in the scenario with dense population. To deal with this problem, cloud-edge interoperation frameworks have been proposed to provision scalable services [18,19].

In this paper, the cloud-assisted multi-cloudlet framework is introduced, as shown in Figure 1. A cloudlet is a multi-core computer or a cluster of well-connected computers [5]. Each sensor is able to offload its computation task to a cloudlet via a wireless AP, or to the remote cloud through the

base station and Internet core. Due to the constrained computation resources of cloudlets and limited communication resources of wireless APs, the computation offloading decision of one sensor can have a significant influence on the processing delay and energy consumption of other sensors with identical computation offloading decisions. As processing delay and energy consumption of computation tasks are critical quality of service (QoS) metrics of IoT sensors, coordinating the computation offloading decisions of different IoT sensors is of vital importance.

This paper investigates the computation offloading decision problem of IoT sensors in the cloud-assisted multi-cloudlet framework. Many existing works have focused on computation offloading problems, including computation offloading problems of mobile devices [20–23], mobile data traffic offloading problems [24,25] and service migration problems [26]. Solving the computation offloading problem of IoT sensors in the cloud-assisted multi-cloudlet framework is challenging. First, IoT sensors are owned by different users with diversified interests. As IoT sensors with identical computation offloading decisions interact with each other, how to coordinate the computation offloading decisions of different sensors is challenging. This work solves the computation offloading decision problem of IoT sensors based on game theory, which is widely adopted to analyze systems with multiple selfish individuals [27]. Another challenge is that the cloud-assisted multi-cloud framework has heterogeneous computation and communication resources. Cloud resources can be provided on demand while computation resources of cloudlets are constrained. Each time one sensor changes its computation offloading decision, the computation and communication resources obtained by this sensor are changed simultaneously. Furthermore, due to the heterogeneous mobile networks (i.e., wireless access network and cellular network) in this framework, IoT sensors can have distinct energy consumption models when making different computation offloading decisions.

In this work, the computation offloading decision problem of IoT sensors is analyzed based on game theory. In more detail, the computation offloading decision problem of IoT sensors is formulated as a computation offloading game. To analyze the property of this game, the tool of potential game is introduced. Based on the definition of potential game, the condition of Nash equilibrium is derived, and the Computation Offloading Decision (COD) algorithm is further designed by leveraging the finite improvement property of potential game. In the COD algorithm, the improvement range is optimized in each improvement iteration, such that the computation complexity of the algorithm is significantly reduced. We implement extensive simulations to evaluate the COD algorithm. Simulation results demonstrate that the COD algorithm can effectively reduce the system cost (defined as the average weighted sum of processing delay and energy consumption of all sensors), compared with the random-selection algorithm and the cloud-first algorithm. Furthermore, the COD algorithm can scale well with increasing IoT sensors.

This paper is organized as follows. Section 2 presents the system model and Section 3 provides problem formulation and analysis. Algorithm design is presented in Section 4 and simulation results are illustrated and analyzed in Section 5. Finally, this work is concluded in Section 6.

2. System Model

This section presents the system model. As shown in Figure 1, there are a set of $\Gamma = \{1, 2, ..., N\}$ IoT sensors, each of which has a computation-intensive delay-sensitive computation task. Consider K cloudlets, to which IoT sensors can offload the computation tasks through wireless access points (APs). Since processing delay and energy consumption are two important factors that influence the performance of IoT sensors, this work considers these two issues as the metrics of QoS. In the cloudlet-based mobile edge computing framework, computation resources of each cloudlet are limited. In addition, IoT sensors connected to the identical wireless APs interfere with each other. Therefore, the performance of one sensor can be affected by the other sensors with identical computation offloading decisions. As both the computation process and the communication process play a significant role in computation offloading of IoT sensors, analysis of these two processes is provided as follows.

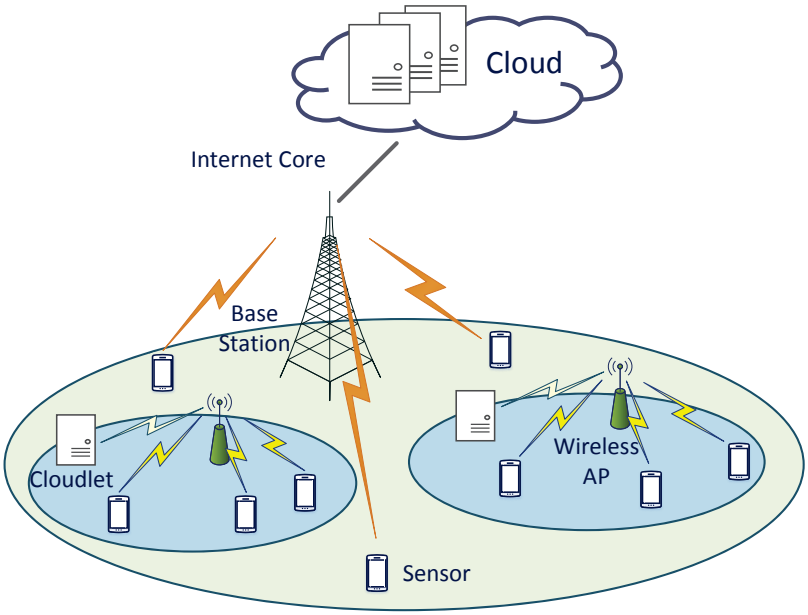


Figure 1. Cloud-assisted cloudlet-based mobile edge computing

2.1. Computation Process

In the cloud-assisted cloudlet-based mobile edge computing framework, each IoT sensor n has a computation task M_n, which is defined as

$$M_n \triangleq \{d_n, u_n\}, \tag{1}$$

where d_n represents the data block transmitted during computation offloading and u_n represents the computation requirements of the computation task. Denote by $\mathbf{s} = (s_1, s_2, ..., s_N)$ the computation offloading decision profile of sensors. Considering K cloudlets, then $s_n = i$ ($i \in \{1, 2, ..., K\}$) when the computation task of sensor n is offloaded to cloudlet i, and $s_n = 0$ if sensor n offloads the computation task to the cloud. Denote by $\mathbf{S_n}$ the strategy set of sensor n.

As scalable computation resources can be provided by the cloud, when the computation task is offloaded to the cloud, the computation rate of each sensor can be guaranteed, denoted as f_n^{c}. Denote by f^{e} the computation rate of each cloudlet and v_n the ratio of computation resources that sensor n can obtain when offloading computation task to the cloudlet. Then, the computation rate of sensor n can be represented as

$$r_n^{\mathrm{comp}}(\mathbf{s}) = \begin{cases} f^{\mathrm{e}} \dfrac{v_n}{\sum\limits_{m \in \Gamma: s_m = s_n}^{N} v_m} & s_n > 0 \\ f_n^{\mathrm{c}} & s_n = 0. \end{cases} \tag{2}$$

Thus, the computation delay of sensor n is

$$t_n^{\mathrm{comp}}(\mathbf{s}) = \frac{u_n}{r_n^{\mathrm{comp}}(\mathbf{s})}. \tag{3}$$

2.2. Communication Process

As shown in Figure 1, each IoT sensor can offload computation tasks to a cloudlet via a wireless AP. Consider the IoT sensors that offload computation tasks to the identical cloudlet via the wireless AP. The spectrum of this wireless AP is shared among these sensors. In this paper, the CSMA access control protocol in [28] is adopted to distribute the spectrum among these sensors. In this access control

protocol, sensors connected to the wireless AP compete for the channel to transmit the computation tasks. Once one sensor captures this channel, the computation task of this sensor is transmitted in the entirety of the channel. Thus, the expected transmission rate of an IoT sensor can be computed as

$$R_n^{\mathrm{e}}(\mathbf{s}) = b_n \frac{w_n}{\sum\limits_{m \in \Gamma : s_m = s_n} w_m}, \tag{4}$$

where b_n is the transmission rate that sensor n can achieve when capturing this channel, and w_n represents the probability of obtaining the channel.

As computation and communication resources of cloudlets are constrained, when cloudlets have dense population, IoT sensors can connect to the Internet core via the base station and further offload computation tasks to the cloud. With the development of cellular network, especially the rapid emergence and deployment of 5G network, sufficient spectrum is provided and extreme low latency is guaranteed. Thus, interference between IoT sensors are not considered when transmitting via the base station. According to the above analysis, the communication delay of sensor n can be summarized as

$$t_n^{\mathrm{comm}}(\mathbf{s}) = \begin{cases} \frac{d_n}{R_n^{\mathrm{e}}(\mathbf{s})}, & s_n > 0, \\ \frac{d_n}{R_n^{\mathrm{BS}}} + t_{\mathrm{RTT}}, & s_n = 0. \end{cases} \tag{5}$$

Here, R_n^{BS} represents the transmission rate of sensor n in the cellular network, and t_{RTT} is the round trip delay between the base station and the cloud.

2.3. Delay and Energy Model

Processing delay and energy consumption are two important QoS metrics of mobile applications. Processing delay consists of computation delay and communication delay, which is given as

$$t_n(\mathbf{s}) = t_n^{\mathrm{comp}}(\mathbf{s}) + t_n^{\mathrm{comm}}(\mathbf{s}). \tag{6}$$

During the computation offloading of IoT sensors, energy consumption only contains the energy consumed to offload computation tasks to cloudlets or to the cloud. When offloading computation tasks to cloudlets, IoT sensors transmit data blocks via wireless APs. This work adopts the energy model similar to the energy model in [29]. The energy consumption when offloading to cloudlets e_n^{e} contains three parts:

- scanning energy e_n^{s}: energy consumption during scanning available APs,
- transmission energy e_n^{t}: energy consumption during data transmission,
- maintaining energy e_n^{m}: energy consumed to maintain interfaces on during computation offloading.

Scanning energy e_n^{s} can be represented by a constant. Transmission energy e_n^{t} is given by

$$e_n^{\mathrm{t}} = a_0 d_n, \tag{7}$$

where a_0 is the energy consumed to transfer one unit of data via wireless APs. Maintaining energy e_n^{m} is computed as

$$e_n^{\mathrm{m}}(\mathbf{s}) = p_n^{\mathrm{m}} \cdot t_n^{\mathrm{comm}}(\mathbf{s}). \tag{8}$$

Here, p_n^{m} is the transmission power of sensor n. Therefore, the energy consumption when offloading to cloudlets ($s_n \in \{1, 2, ..., K\}$) is

$$e_n(\mathbf{s}) = e_n^{\mathrm{s}} + a_0 d_n + p_n^{\mathrm{m}} \cdot t_n^{\mathrm{comm}}(\mathbf{s}). \tag{9}$$

When offloading computation tasks to the cloud through the base station and Internet core, the energy consumption of IoT sensors during computation offloading contains includes tail energy e_n^{tail} as well. Thus, the energy consumption when offloading to the cloud ($s_n = 0$) is

$$e_n(\mathbf{s}) = e_n^{\text{s}'} + e_n^{\text{tail}} + a_1 d_n + p_n^{\text{m}'} \cdot t_n^{\text{comm}}(\mathbf{s}), \tag{10}$$

where $e_n^{\text{s}'}$ and $p_n^{\text{m}'}$ are scanning energy and maintaining power, respectively, when offloading to the cloud, and a_1 is the energy consumed to transfer one unit of data via the base station.

3. Problem Formulation and Analysis

In the cloud-assisted multi-cloudlet framework, IoT sensors contend for the computation and communication resources of the cloudlets and wireless APs to meet the QoS requirements of IoT applications. Since IoT sensors are owned by different individuals that pursue diversified interests, decentralized computation offloading decisions should be made to mimic the selfish property of the sensor users. As systems consisting of multiple selfish individuals are usually analyzed based on game theory [27], this work formulates the computation offloading problem of multiple IoT sensors as a computation offloading game. The details are presented as follows.

3.1. Computation Offloading Game

In the cloud-assisted multi-cloudlet framework, the computation offloading problem of IoT sensors is formulated as a computation offloading game. The players are the IoT sensors that have computation tasks to offload. The strategy of each player n ($n \in \Gamma$) is the computation offloading decision s_n ($s_n \in \{0, 1, ..., K\}$). Denote by s_{-n} the computation offloading decisions of all IoT sensors except n, i.e.,

$$s_{-n} \triangleq \{s_1, ...s_{n-1}, s_{n+1}, ..., s_N\}. \tag{11}$$

For each IoT sensor, the processing delay and energy consumption of the computation task are important QoS metrics, thus the cost function of IoT sensor n is defined as

$$C_n(\mathbf{s}) = C_n(s_n, s_{-n}) = t_n(\mathbf{s}) + \lambda_n \cdot e_n(\mathbf{s}). \tag{12}$$

Here, λ_n represents the weight parameter of energy consumption over processing delay, which is assigned by sensor n based on its interest.

According to the results in Section 2, when sensor n offloads the computation task to a cloudlet via a wireless AP (i.e., $s_n \in \{1, 2, ..., K\}$), the cost function of sensor n can be represented as

$$\begin{aligned} C_n(\mathbf{s}) &= \left(\frac{u_n \sum\limits_{m\in\Gamma:s_m=s_n}^{N} v_m}{f^{\text{e}} v_n} + \frac{d_n \sum\limits_{m\in\Gamma:s_m=s_n} w_m}{b_n w_n}\right) + \lambda_n \cdot \left(e_n^{\text{s}} + a_0 d_n + p_n^{\text{m}} \cdot \frac{d_n \sum\limits_{m\in\Gamma:s_m=s_n} w_m}{b_n w_n}\right) \\ &= \frac{u_n}{f^{\text{e}}} \cdot \frac{\sum\limits_{m\in\Gamma:s_m=s_n}^{N} v_m}{v_n} + \frac{d_n}{b_n}(1 + \lambda_n p_n^{\text{m}}) \cdot \frac{\sum\limits_{m\in\Gamma:s_m=s_n} w_m}{w_n} + (\lambda_n e_n^{\text{s}} + \lambda_n a_0 d_n). \end{aligned} \tag{13}$$

When sensor n offloads the computation task to the cloud (i.e., $s_n = 0$), the cost function is

$$C_n^0 = \left(\frac{u_n}{f_n^{\text{c}}} + \frac{d_n}{R_n^{\text{BS}}} + t_{\text{RTT}}\right) + \lambda_n \cdot \left[e_n^{\text{s}'} + e_n^{\text{tail}} + a_1 d_n + p_n^{\text{m}'} \cdot \left(\frac{d_n}{R_n^{\text{BS}}} + t_{\text{RTT}}\right)\right]. \tag{14}$$

3.2. Analysis

Based on game theory, a Nash equilibrium of a game is defined as in Definition 1 [27].

Definition 1. *A strategy profile* $s^*=\{s_1^*, s_2^*, ..., s_n^*\}$ *is a Nash equilibrium, if for any* $n \in \Gamma$ *and any* $s_n \in \{0, 1, ..., K\}$, *it is satisfied that*

$$C_n(s_n^*, s_{-n}) \leq C_n(s_n, s_{-n}). \tag{15}$$

According to Definition 1, when the computation offloading game of IoT sensors reaches Nash equilibrium, the strategy of each player is the best computation offloading decision that minimizes its cost function, given the strategies of all the other players. Thus, no player has the motivation to violate the current computation offloading decision.

For the computation offloading game of IoT sensors, each IoT sensor chooses the computation offloading strategy to minimize its cost function selfishly. Since the sensors offloading to the identical cloudlet via the wireless AP contend for the computation and communication resources, when one sensor n changes its strategy from s_n to $s_n^{'}$, the cost of sensor m that has the strategy of $s_n^{'}$ (i.e., $s_m = s_n^{'}$) is influenced. Denote by s_m^* the current computation offloading decision of sensor m and define Δ_m as

$$\Delta_m = \min_{s_m \neq s_m^*} c_m(s_m, s_{-m}) - c_m(s_m^*, s_{-m}). \tag{16}$$

Then, Δ_m represents the smallest difference between all the other strategies and the current strategy of sensor m. Denote by m^* the sensor that has the smallest Δ_m, i.e.,

$$m^* = \arg\min_{s_m = s_n^{'}} \Delta_m \,. \tag{17}$$

When sensor n changes its strategy from s_n to $s_n^{'}$, the costs of sensors that offload computation tasks to $s_n^{'}$ increase due to the reduced computation and communication resources allocated to these sensors. It is intuitive that sensor m^* is most likely to change its current computation offloading strategy, and the strategy is only changed when new sensors with larger Δ choose to offload the computation tasks to s_m (i.e., $s_n^{'}$). When sensor m^* chooses a new computation offloading strategy $s_m^{'}$, it may further motivate sensors with the computation offloading strategy of $s_m^{'}$ and smaller Δ to change the computation offloading strategies.

According to the above analysis, when one sensor changes its computation offloading strategy, a chain reaction of strategy changes can be incurred. In the following work, we further explore the properties of the computation offloading game and analyze the existence of Nash equilibrium.

4. Algorithm Design

This section first derives the condition of Nash equilibrium by introducing the concept of a potential game [30]. Then, the Computation Offloading Decision (COD) algorithm is designed by exploiting the property of the computation offloading game.

4.1. Condition of Nash Equilibrium

According to [30], a potential game is defined as follows:

Definition 2. *The computation offloading game of IoT sensors is a potential game if there exists a function* $\Theta(\mathbf{s})$ *such that if for any* n, $s_n, s_n^{'} \in \mathbf{S_n}$ *and* $s_{-n} \in \prod_{m \in \Gamma \setminus \{n\}} \mathbf{S_n}$, *if*

$$C_n(s_n, s_{-n}) < C_n(s_n^{'}, s_{-n}), \tag{18}$$

there is

$$\Theta(s_n, s_{-n}) < \Theta(s_n^{'}, s_{-n}). \tag{19}$$

The function $\Theta(\mathbf{s})$ *is called potential function.*

It is indicated from Definition 2 that in a potential game, when any IoT sensor $n \in \Gamma$ changes its computation offloading decision from $s_n^{'}$ to s_n so as to reduce its cost function, the value of the potential function also decreases. When the potential game reaches a Nash equilibrium, the potential function can not be further reduced. It has been proved that any finite potential game has a Nash equilibrium [30]. Therefore, if we want to derive the condition of Nash equilibrium in the computation offloading game of IoT sensors, we simply need to investigate the condition of a potential game.

Theorem 1. *The computation offloading game of IoT sensors is a potential game if for every sensor $n \in \Gamma$, there is*

$$\frac{w_n}{\sum\limits_{m\in\Gamma:s_m=s_n}^{N} w_m} = \frac{v_n}{\sum\limits_{m\in\Gamma:s_m=s_n}^{N} v_m}. \tag{20}$$

Proof. Substitute Equation (20) into Equation (13), and it is derived that when sensor n offloads the computation task to a cloudlet (i.e., $s_n \in \{1, 2, ..., K\}$), the cost function is

$$C_n(\mathbf{s}) = [\frac{u_n}{f^{\mathrm{e}}} + \frac{d_n}{b_n}(1 + \lambda_n p_n^{\mathrm{m}})] \cdot \frac{\sum\limits_{m\in\Gamma:s_m=s_n} w_m}{w_n} + (\lambda_n e_n^{\mathrm{s}} + \lambda_n a_0 d_n). \tag{21}$$

Define Ω_n as

$$\Omega_n = w_n[C_n^0 - (\lambda_n e_n^{\mathrm{s}} + \lambda_n a_0 d_n)] / [\frac{u_n}{f^{\mathrm{e}}} + \frac{d_n}{b_n}(1 + \lambda_n p_n^{\mathrm{m}})], \tag{22}$$

where C_n^0 is represented in Equation (14). Then, it can be proved that there exists a potential function $\Theta(\mathbf{s})$

$$\Theta(\mathbf{s}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} w_n w_m f(s_m = s_n) f(s_n > 0) + \sum_{n=1}^{N} \Omega_n w_n f(s_n = 0) \tag{23}$$

in two cases:

Case 1: if $s_n > 0, s_n^{'} > 0$, then

$$C_n(s_n, s_{-n}) - C_n(s_n^{'}, s_{-n}) = [\frac{u_n}{f^{\mathrm{e}}} + \frac{d_n}{b_n}(1 + \lambda_n p_n^{\mathrm{m}})] \cdot \frac{\sum\limits_{m\in\Gamma:s_m=s_n} w_m - \sum\limits_{l\in\Gamma:s_m=s_n^{'}}, w_l}{w_n} \tag{24}$$

and

$$\begin{aligned} \Theta_n(s_n, s_{-n}) - \Theta_n(s_n^{'}, s_{-n}) &= w_n(\sum_{m\in\Gamma:s_m=s_n} w_m - \sum_{l\in\Gamma:s_m=s_n^{'}} w_l) \\ &= \frac{(w_n)^2}{[\frac{u_n}{f^{\mathrm{e}}} + \frac{d_n}{b_n}(1 + \lambda_n p_n^{\mathrm{m}})]}[C_n(s_n, s_{-n}) - C_n(s_n^{'}, s_{-n})]. \end{aligned} \tag{25}$$

In this case, if $C_n(s_n, s_{-n}) - C_n(s_n^{'}, s_{-n}) > 0$, it is satisfied that $\Theta_n(s_n, s_{-n}) - \Theta_n(s_n^{'}, s_{-n}) > 0$.

Case 2: if $s_n > 0, s_n^{'} = 0$, then

$$C_n(s_n, s_{-n}) - C_n(s_n^{'}, s_{-n}) = \frac{u_n}{f^{\mathrm{e}}} + \frac{d_n}{b_n}(1 + \lambda_n p_n^{\mathrm{m}})] \cdot \frac{\sum\limits_{m\in\Gamma:s_m=s_n} w_m}{w_n} + (\lambda_n e_n^{\mathrm{s}} + \lambda_n a_0 d_n) - C_n^0, \tag{26}$$

and

$$\begin{aligned}\Theta_n(s_n, s_{-n}) - \Theta_n(s_n^{'}, s_{-n}) &= w_n\Big(\sum_{m\in\Gamma:s_m=s_n} w_m - \sum_{l\in\Gamma:s_m=s_n^{'}} w_l\Big) \\ &= \frac{(w_n)^2}{[\frac{u_n}{f^e} + \frac{d_n}{b_n}(1+\lambda_n p_n^{\mathrm{m}})]}[C_n(s_n, s_{-n}) - C_n(s_n^{'}, s_{-n})].\end{aligned} \tag{27}$$

In this case, if $C_n(s_n, s_{-n}) - C_n(s_n^{'}, s_{-n}) > 0$, $\Theta_n(s_n, s_{-n}) - \Theta_n(s_n^{'}, s_{-n}) > 0$ is satisfied too.

Therefore, if the condition in Equation (20) is satisfied in the computation offloading game of IoT sensors, the game is a potential game with the potential function presented in Equation (23). □

4.2. Computation Offloading Decision Algorithm

It has been proved that the computation offloading game of IoT sensors is a potential game when the condition in Equation (20) is satisfied. In a potential game, the Nash equilibrium can always be achieved after finite improvement iterations, which is called finite improvement property [30]. This part provides the design of the Computation Offloading Decision (COD) algorithm by exploiting the finite improvement property of the computation offloading game. The details are shown in Algorithm 1.

Denote by ρ_n^t the better strategy set of sensor n in the tth iteration: if $C_n(s_n^{t-1}, s_{-n}) > C_n(k, s_{-n})$, then $k \in \rho_n^t$. d_n^t represents the improvement decision of sensor n in the tth iteration and ξ_n^t is the improvement range of cost function defined as $\xi_n^t = C_n(s_n^{t-1}, s_{-n}) - C_n(d_n^t, s_{-n})$.

Algorithm 1 Computation Offloading Decision (COD) algorithm

Input:
Computation tasks of sensors $M_n(n \in \Gamma)$.
Output:
Computation offloading strategy of sensors $s_n(n \in \Gamma)$.
Initialization
Each sensor initially offloads its computation task to the cloud, $s_n^0 = 0$ $(n \in \Gamma)$.

1: Compute the cost of each sensor n when offloading the computation task to the cloud, C_n^0, based on Equation (14).
2: **for** each improvement iteration t **do**
3: **for** each sensor $n \in \Psi(t-1)$ **do**
4: obtain the computation and communication resources allocated to sensor n when offloading the computation task to each cloudlet according to s_{-n}^{t-1}.
5: compute the better strategy set ρ_n^t based on Equations (13) and (14).
6: choose the improvement decision d_n^t from ρ_n^t to minimize the cost function.
7: **end for**
8: **if** the better strategy set of all sensors $\bigcup_{n=1}^{N} \rho_n^t \neq \emptyset$ **then**
9: each sensor n with $\rho_n^t \neq \emptyset$ competes for the improvement opportunity, and the sensor with larger ξ_n^t is endowed with higher priority of winning the contention.
10: **if** sensor n wins the contention **then**
11: $s_n^t = d_n^t$.
12: **end if**
13: **else**
14: **break**;
15: **end if**
16: **end for**

It is indicated in Definition 2 that the potential function decreases when each player changes its strategy to reduce its cost function. In addition, the computation offloading game of IoT sensors has a finite improvement property. As in each improvement iteration sensors with larger ξ_n^t are endowed with a higher priority of winning the improvement opportunity (as presented in step 9),

the improvement range of the potential function is optimized in each iteration. Therefore, the number of improvement iterations is reduced and the COD algorithm can have good scalability.

5. Simulations and Results

In this section, extensive simulations are conducted to evaluate the COD algorithm. Consider the simulation scenario in which there are N IoT sensors, each of which has a computation task $M_n = \{d_n, u_n\}$ to offload. The simulations take the face recognition application in [31] as the example of the IoT computation tasks. As IoT sensors have heterogeneous computation requests, the data load d_n and computation requirements u_n follow normal distribution in these simulations, with the parameters of (420, 42) KB and (2, 0.2) Giga CPU cycles, respectively. There are 10 cloudlets, each of which has the computation capacity of 24 GHz. IoT sensors can offload computation tasks to each cloudlet via a wireless AP. The transmission rates of these wireless APs follow normal distribution, with the expectation of 5 Mbps and the standard deviation of 1 Mbps. When offloading the computation task to the cloud, a T2.nano instance is provided to serve each sensor, the computation rate of which is 2.4 GHz [32]. The upload rate of each sensor to the base station is 2 Mbps. The round trip delay from the base station to the cloud is 50 ms [33,34]. The energy consumption parameters are based on the measurement results in [29], as shown in Table 1.

Table 1. Energy consumption parameters [29].

	Wireless AP	**Base Station**
Scanning Energy (J)	5.9	3.5
Transmission Power (J/KB)	0.007	0.025
Maintaining Power (W)	0.05	0.02
Tail Energy (J)	0	7.75

The performance of the COD algorithm is compared with the random-selection algorithm and the cloud-first algorithm. In the random-selection algorithm, each IoT sensor randomly selects a computation offloading decision among the cloudlets and the cloud. In the cloud-first algorithm, all IoT sensors offload the computation tasks to the cloud.

5.1. Performance of the COD Algorithm

Defining the system cost as the average cost of all IoT sensors,

$$C^{\text{sys}} = \frac{\sum_{n=1}^{N} C_n(\mathbf{s})}{N}, \tag{28}$$

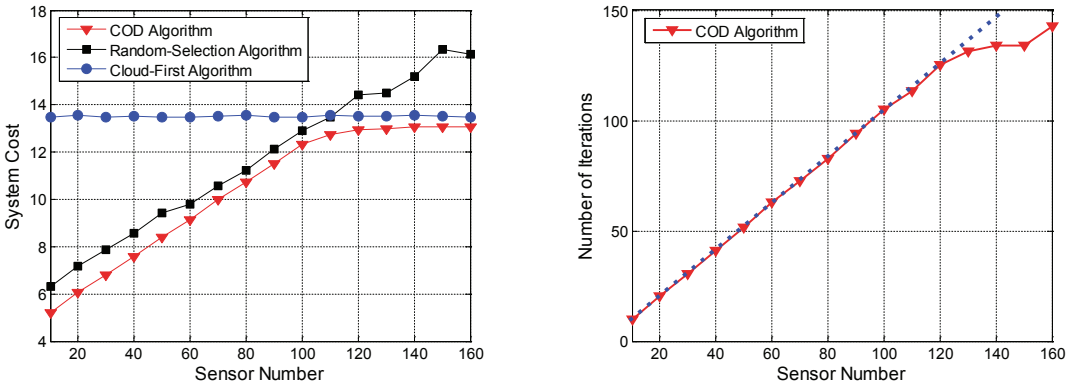
then the system costs of the three algorithms are illustrated as Figure 2a.

As shown in Figure 2a, the COD algorithm can significantly reduce the system cost compared with the random-selection algorithm and the cloud first algorithm. In the cloud-assisted system, the base station provides abundant communication resources and the cloud has scalable computation resources. Thus, the system cost of the cloud-first algorithm remains almost unchanged with increasing sensor number.

When the sensor number is small, the cloudlets can provide sufficient computation resources and the wireless APs have abundant communication resources, offloading computation tasks to cloudlets yields lower costs than offloading to the cloud. In the COD algorithm and the random-selection algorithm, part of IoT sensors offload computation tasks to cloudlets, thus the system costs are lower than the cloud-first algorithm. As sensor number increases, the computation resources of cloudlets and the communication resources of wireless APs become scarce. The system costs of the COD algorithm and the random-selection algorithm increase rapidly with increasing sensor number. In the COD

algorithm, the IoT sensor with the largest improvement range is endowed with the highest priority to update the computation offloading strategy in each iteration. Thus, the system cost of the COD algorithm increases slower than the random-selection algorithm, and finally approaches the result of the cloud-first algorithm.

To evaluate the scalability of the COD algorithm, simulations are further implemented to record the number of improvement iterations with different sensor numbers. The results are shown in Figure 2b. It is indicated that the number of iterations increases approximately linearly with the number of sensors. As revealed in Algorithm 1, the computation complexity of each improvement iteration is O(NK). Therefore, the computation complexity of the COD algorithm approaches O(N^2K), and the COD algorithm scales well with increasing sensor number.

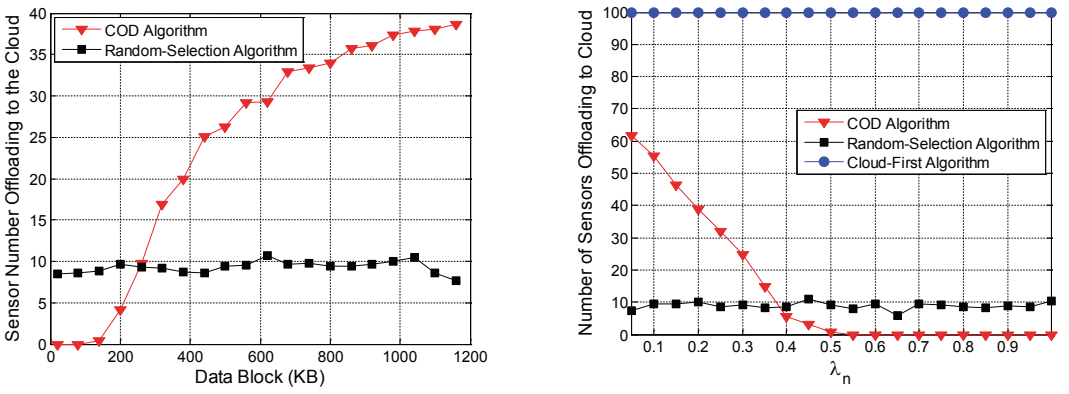


(**a**) System Cost of Different Algorithms (**b**) Scalability of the COD Algorithm

Figure 2. Performance of the COD Algorithm.

5.2. Influence of Different Parameters

This part evaluates the influence of different parameters on the results of the COD algorithm. Simulations are conducted to record the number of sensors offloading to the cloud (the overall number of sensors n = 100). The results are illustrated as Figure 3a,b.



(**a**) Influence of data block (**b**) Influence of λ_n

Figure 3. Influence of different parameters.

Figure 3a presents the influence of the size of trasmitted data block during computation offloading. In the random-selection algorithm, the number of sensors offloading to the cloud almost remains unchanged, since computation offloading strategies of sensors are selected randomly. As the

communication resources of wireless APs are limited, the costs of sensors increase with growing data block when offloading the computation tasks to cloudlets. Therefore, increasing sensors offload the computation tasks to the cloud in the COD algorithm.

Figure 3b illustrates the influence of weight parameter of energy consumption over processing delay (λ_n). In this simulation, the transmitted data block is 420 KB, and the sensor number n = 100. According to [29], the energy consumption when transmitting the data block via the base station is higher than transmitting via wireless APs. As λ_n increases, energy consumption occupies more in the cost of sensors. Thus, in the COD algorithm, the number of sensors offloading to the cloud (via the base station) decreases with increasing λ_n.

6. Conclusions

This work has investigated the computation offloading decision problem of IoT sensors in the cloud-assisted multi-cloudlet framework. Theoretic analysis has been presented and the condition of Nash equilibrium has been derived based on the potential game. By exploiting the finite property of the computation offloading game of IoT sensors, the COD algorithm has been proposed. Simulation results have demonstrated that the COD algorithm can effectively reduce the system cost compared with the random-selection algorithm and the cloud-first algorithm. In addition, the complexity of the COD algorithm is O(N^2K), thus this algorithm can scale well with increasing IoT sensors.

Author Contributions: Conceptualization, X.M.; Formal analysis, X.M.; Investigation, X.M., H.Z. and J.L.; Methodology, X.M.; Resources, C.L. and J.L.; Software, X.M. and H.Z.; Supervision, C.L.

Funding: This research was funded by the National Natural Science Foundation of China Grant No. 61472199.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ashton, K. That 'Internet of Things' Thing. *RFID J.* **2009**. *22*, 97–114.
2. Vilmo, A.; Medaglia, C.; Moroni, A. Vision and Challenges for Realising the Internet of Things. *Hot Work. Technol.* **2010**, *35*, 59–60.
3. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
4. Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper. Available online: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html (accessed on 11 June 2018).
5. Satyanarayanan, M.; Bahl, P.; Caceres, R.; Davies, N. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Comput.* **2009**, *8*, 14–23. [CrossRef]
6. Luan, T.H.; Gao, L.; Li, Z.; Xiang, Y.; Wei, G.; Sun, L. Fog computing: Focusing on mobile users at the edge. *arXiv* **2015**, arXiv:1502.01815.
7. Osanaiye, O.; Chen, S.; Yan, Z.; Lu, R.; Choo, K.K.R.; Dlodlo, M. From cloud to fog computing: A review and a conceptual live VM migration framework. *IEEE Access* **2017**, *5*, 8284–8300. [CrossRef]
8. ETSI. *Mobile Edge Computing (MEC): Framework and Reference Architecture*; ETSI: Sophia Antipolis CEDEX, France, 2016.
9. Roman, R.; Lopez, J.; Mambo, M. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Gener. Comput. Syst.* **2018**, *78*, 680–698. [CrossRef]
10. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. Mobile edge computing: Survey and research outlook. *arXiv* **2017**, arXiv:1701.01090.
11. Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1628–1656. [CrossRef]
12. Li, H.; Ota, K.; Dong, M. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. *IEEE Netw.* **2018**, *32*, 96–101. [CrossRef]
13. Xu, J.; Ota, K.; Dong, M. Saving Energy on the Edge: In-Memory Caching for Multi-Tier Heterogeneous Networks. *IEEE Commun. Mag.* **2018**, *56*, 102–107. [CrossRef]

14. Zhang, S.; Zhang, N.; Yang, P.; Shen, X. Cost-Effective Cache Deployment in Mobile Heterogeneous Networks. *IEEE Trans. Veh. Technol.* **2017**, *66*, 11264–11276. [CrossRef]
15. Zhang, S.; He, P.; Suto, K.; Yang, P.; Zhao, L.; Shen, X.S. Cooperative Edge Caching in User-Centric Clustered Mobile Networks. *IEEE Trans. Mob. Comput.* **2017**. [CrossRef]
16. Wu, J.; Dong, M.; Ota, K.; Li, J.; Guan, Z. FCSS: Fog computing based content-aware filtering for security services in information centric social networks. *IEEE Trans. Emerg. Top. Comput.* **2017**. [CrossRef]
17. Zhang, S.; Zhang, N.; Fang, X.; Yang, P.; Shen, X. Self-Sustaining Caching Stations: Toward Cost-Effective 5G-Enabled Vehicular Networks. *IEEE Commun. Mag.* **2017**, *55*, 202–208. [CrossRef]
18. Yang, P.; Zhang, N.; Bi, Y.; Yu, L.; Shen, X. Catalyzing cloud-fog interoperation in 5G wireless networks: An SDN approach. *IEEE Netw.* **2017**, *31*, 14–20. [CrossRef]
19. Ma, X.; Zhang, S.; Li, W.; Zhang, P.; Lin, C.; Shen, X. Cost-Efficient Workload Scheduling in Cloud Assisted Mobile Edge Computing. In Proceedings of the IEEE/ACM International Symposium on Quality of Service (IWQoS'17), Vilanova i la Geltru, Spain, 14–16 June 2017; pp. 1–10.
20. Tao, X.; Ota, K.; Dong, M.; Qi, H.; Li, K. Performance guaranteed computation offloading for mobile-edge cloud computing. *IEEE Wirel. Commun. Lett.* **2017**, *6*, 774–777. [CrossRef]
21. Ma, X.; Lin, C.; Xiang, X.; Chen, C. Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing. In Proceedings of the ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'15), Cancum, Mexico, 2–6 November 2015; pp. 271–278.
22. Chen, X. Decentralized computation offloading game for mobile cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 974–983. [CrossRef]
23. Chen, X.; Jiao, L.; Li, W.; Fu, X. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Netw.* **2016**, *24*, 2795–2808. [CrossRef]
24. Lei, T.; Wang, S.; Li, J.; Yang, F. AOM: Adaptive mobile data traffic offloading for M2M networks. *Pers. Ubiquitous Comput.* **2016**, *20*, 863–873. [CrossRef]
25. Wang, S.; Lei, T.; Zhang, L.; Hsu, C.H.; Yang, F. Offloading mobile data traffic for QoS-aware service provision in vehicular cyber-physical systems. *Future Gener. Comput. Syst.* **2016**, *61*, 118–127. [CrossRef]
26. Wang, S.; Xu, J.; Zhang, N.; Liu, Y. A Survey on Service Migration in Mobile Edge Computing. *IEEE Access* **2018**, *6*, 23511–23528. [CrossRef]
27. Osborne, M.J.; Rubinstein, A. *A Course in Game Theory*; MIT Press: Cambridge, MA, USA, 1994.
28. Standard, F. *Telecommunications: Glossary of Telecommunication Terms;* General Services Administration: Washington, D.C., USA, 1996; Volume 7.
29. Balasubramanian, N.; Balasubramanian, A.; Venkataramani, A. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In Proceedings of the ACM SIGCOMM Conference on Internet Measurement, Chicago, IL, USA, 4–6 November 2009; pp. 280–293.
30. Monderer, D.; Shapley, L.S. Potential games. *Games Econ. Behav.* **1996**, *14*, 124–143. [CrossRef]
31. Soyata, T.; Muraleedharan, R.; Funai, C.; Kwon, M.; Heinzelman, W. Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In Proceedings of the IEEE Symposium on Computers and Communications (ISCC'12), Cappadocia, Turkey, 1–4 July 2012; pp. 59–66.
32. Amazon. EC2 Instance. Available online: https://amazonaws-china.com/cn/ec2/instance-types/ (accessed on 12 June 2018).
33. Li, A.; Yang, X.; Kandula, S.; M, Z. CloudCmp: Comparing Public Cloud Providers. In Proceedings of the ACM Internet Measurement Conference (IMC'10), Melbourne, Australia, 1 November 2010; pp. 1–14.
34. Claypool, M.; Claypool, K. Latency and player actions in online games. *Commun. ACM* **2006**, *49*, 40–45. [CrossRef]