IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 11, 2022, accepted March 12, 2022, date of publication March 16, 2022, date of current version March 24, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3160198

Balanced Computing Offloading for Selfish IoT Devices in Fog Computing

SUN YU-JIE, WANG HUI, AND ZHANG CHENG-XIANG
School of Mathematics and Computer Science, Zhejiang Normal University, Jinhua 321000, China
Corresponding author: Wang Hui (hwang@zjnu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 62171413.

ABSTRACT Fog computing, which provides low-latency computing services at the network edge, is an enabler for the emerging Internet of Things (IoT) systems. Offloading tasks to the fog that is closer to IoT users for processing has become a means to ensure that tasks are completed quickly. Fog computing cannot only reduce the congestion of the backbone network but also ensure that the task is completed within the specified time. Since fog resources are limited, there will be resource competition among IoT devices. How to quickly and efficiently make an optimal computation offloading decision for individual selfish IoT devices is a fundamental research issue. This article regards the process of multiple IoT devices competing for fog devices as a game and proposes a distributed computation offloading algorithm. The goal is to optimize the balance of computation delay, energy consumption, and cost for fog nodes. The competition between IoT nodes eventually reaches an equilibrium point, that is the Nash equilibrium point. We prove the existence of Nash equilibrium by Weighted Potential Game. In addition, if a large number of IoT devices select the same node for offloading, which will cause the fog node to run out of power and make some networks unable to work normally. Further, causing part of the network to be paralyzed. Therefore, the paper considers the fairness of offloading to extend the network life cycle. A calculation rate adjustment algorithm is designed for the fairness of offloading to ensure that fog nodes do not run out of power and fail. This paper not only fully considers the performance of the IoT device, but also considers the fairness of the fog. Numerous experiments proved the effectiveness of the proposed algorithm.

INDEX TERMS Internet of Things, fog computing, computation offloading, Nash equilibrium.

I. INTRODUCTION

International Data Corporation (IDC) predicts that the number of sensors connected to the network will increase to 30 billion, and the number of connected devices will increase from 50 billion to 1 trillion by 2022. All these devices are connected to the network and construct the Internet of Things (IoT) systems [1]. As the increasing number of computation-intensive applications (e.g., augmented reality and face recognition) appears, higher requirements of computing power are placed on IoT devices. The IoT devices with limited memory and computing power cannot handle computation-intensive applications effectively. How to deal with the data generated by a large number of devices has become a problem. It is inefficient to offload tasks to the cloud data center because this will cause network bandwidth overhead, as much of the data can be filtered due to high redundancy. Besides, the long distance between the IoT devices and the cloud data center also lead to unacceptable task processing delay, huge transmission energy consumption, poor support to mobility, and problems of security[2]–[5]. Therefore, fog computing is proposed. Fog servers are a cloud of servers close to the ground or the edge. Cisco proposed the concept of fog computing, which introduced fog as a cloud near the ground or edge [6], [7]. Not only can fog computing handle low-latency tasks but also effectively reduce network congestion [8], [9]. As a supplement to cloud computing, fog computing extends the functions of cloud computing to the edge[10]. Fog computing consists of a large number of geographically distributed fog servers. Compared with the cloud, the computing power and storage capacity of fog nodes are very limited[11], [12]. Therefore, how to effectively distribute fog resources is a very important but problematic issue [13]. There is currently no uniform method. Some researchers adopt a centralized resource allocation method and can get a better allocation

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

method. Nevertheless, selfish IoT users are interested in optimizing their quality of experience (QoE) individually. They may not follow the strategies that aim to optimize the overall system performance. This will inevitably cause competition between IoT devices. So this paper we design a distributed computation offloading method for delay-sensitive tasks. This article regards IoT device competition with fog resources as a game. This article regards the competition of IoT devices with fog resources as a game. Each device makes the most beneficial computing offloading decision based on the choices of other devices. And prove that this game will eventually reach a Nash equilibrium point. Task offloading schemes focusing on minimizing the computation delay or total energy consumption in existing literature may lead to extremely heavy burdens on the fog nodes that are close to the fog nodes or have high processing capabilities, which will result in the death of some important fog nodes and even serious network problems. A large number of IoT devices select the same node for offloading, which will cause the fog node to run out of power and make some networks unable to work normally. So we design a computation rate adjustment(CRD) algorithm to reduce the unfairness of offloading. The contributions of this paper can be summarized as follows.

- This paper solves a multi-objective optimization problem which concludes the task processing delay, energy consumption, and the cost of offloading.
- This article regards the competition of IoT devices with fog resources as a game. Each IoT device tries to minimize its objective function. And prove that this game will eventually reach a Nash equilibrium point by Weighted Potential Game [14]. Based on the game, this paper design offloading algorithm(DTO) and ε Distributed task offloading algorithm(ε-DTO). ε Distributed task offloading algorithm convergence speed is fast, but the optimization effect is reduced.
- To prevent the failure of fog nodes and affect the operation of the network. And fairness is important for extending the network lifetime [16], [28]. The computational offloading in this paper takes into account the fairness of offloading. This paper proposed a computation rate adjustment(CRD) algorithm to reduce the unfairness of offloading.

II. RELATED WORKS

In this section, we survey the existing literature on computation offloading. Offloading is not a trivial issue in fog computing. A large body of recent research worked on addressing the challenges in offloading. The four offload methods were proposed in [15]. 1) Local Mobile Execution, 2) D2D Offloaded Execution, 3) Direct Fog Offloaded Execution, 4) D2D-Assisted Fog Offloaded Execution. This article mainly considers two offloading methods 1 and 3. In [16], the tasks are divided into three categories: 1) hard-deadline-based tasks 2) soft-deadline-based tasks and 3) no-deadline-based tasks. In this paper, we mainly focus on hard-deadline-based tasks. Hard-deadline-based tasks mean that if the processing time of tasks exceeds the deadline, it is failed. It is also called a delay-sensitive task. In this paper, we mainly focus on latency-sensitive tasks. From the algorithm point of view, the current articles are divided into two categories. One is the approximate algorithm and the other is the meta-heuristic algorithm.

In [17] The Gale-Shapley (GS) algorithm is applied to reach a stable matching to achieve many-to-many computation offloading. However, this algorithm needs a central node to collect all information from edge nodes to make a decision. Once the central node collapses, the algorithm will collapse. Artificial Neural Networks (ANN) were used to predict the offloading time and find the optimal device to offload in [18]. The fog server sends the trained model to the edge node. This paper does not take into account the size of the ANN model and it mainly optimizes the calculation delay. In [19], The interior point method is used to optimize the task calculation delay and calculation energy consumption at the same time to obtain the optimal offloading power and the appropriate offloading task size. This article only considers a single fog node and does not consider the case of multiple fog nodes.

In addition to the myopia algorithm proposed in the above article, a large number of articles use heuristic algorithms for computational offloading. In [20], two nature-inspired meta-heuristic schedulers, namely ant colony optimization (ACO) and particle swarm optimization (PSO) are used to propose two different scheduling algorithms to make an optimal decision. In [21], it transforms the non-convex problem into a convex one to minimize energy consumption under the latency constraint and finite MEC computation capacity. and apply convex optimization to solve it. It assumes that the equipment is non-selfish and will follow the overall goal of minimizing the decision. However, in most cases, the equipment is selfish and only considers the maximization of its interests. In [22], this work first investigates a MEC system consisting of mobile devices and heterogeneous edge servers that support various radio access technologies. An optimal offloading node selection strategy is formulated as a Markov decision process (MDP) and solved by employing the value iteration algorithm (VIA). However, this article only focuses on time. In [23], it formulates the problem into a multiobjective model with two scheduling objectives, involving deployment cost and service latency. multi-replicas Pareto ant colony optimization (MRPACO) is proposed to solve the offloading problem. The weighted total cost combines delay and energy consumption is taken as the optimization goal. First, a reinforcement learning algorithm Q-learning based on the Markov decision process is proposed to solve the problem for minimizing weighted total cost [24].

From a structural point of view, the current computing offloading methods are divided into two types, one is a centralized algorithm, the other is a distributed algorithm. The centralized algorithm can get better optimization results, but the algorithm complexity is high and the number of communications is large. Such as [17], [18], [21] is a

centralized algorithm. In [25], [26], [29], they proposed the distributed computation offloading algorithm. And they proved the Nash equilibrium through the potential game or weighted potential game. The goal of optimization is not considered comprehensive. At the same time, the problem of offloading balance is not considered. This article also considers the task processing delay, energy consumption, and the price of the fog node. At the same time, the offloading balance is considered based on the idea of maximizing the network life cycle. In [30], these two articles talked about the fairness of offloading, which can delay the network life cycle. Their focus is on maximizing the network life cycle. In [31]–[33], they apply a drift-plus-penalty based Lyapunov optimization approach to efficient provision of both job assignment and resource allocation. These articles also use distributed algorithms but consider long-term performance.

It can be seen that most of the papers mentioned above only consider time delay, energy consumption, or the cost of using fog nodes. This paper considers the task processing delay, calculation energy consumption, and the calculation cost of the fog node. This article formulate the computation offloading as a distributed game to minimize the combination of latency, or energy consumption and offloading cost. In addition, the fairness of offloading is also considered, which is of great significance for maximizing the network life cycle.

The remainder of this article is organized as follows. In Section III, we introduce the system model. In section VI, we formulate the computation offloading as a distributed game. Distributed task offloading algorithm(DTO) and ε Distributed task offloading are proposed to solve the offloading question. computation rate adjustment(CRD) algorithm is proposed to ensure fairness between fog nodes. In section V, we evaluate the performance of the proposed algorithm. Finally, we conclude in section VI.

III. SYSTEM MODEL

We consider an IoT system with a hierarchical computing structure and a set of IoT devices. This article divides the tasks of IoT devices into two types: delay-sensitive and delay-insensitive. This article focuses more on delay-sensitive tasks. The system model consists of three layers, the IoT layer, the fog layer, and the cloud layer, as shown in FIGURE 1. For convenience, the main notations used are summarized in Table 1.

TABLE 1. Major notations.

Notation	Explantation
N	The set of IoT devices.
M	The set of fog nodes.
T_n	Task generated by IoT device n.
L_n	Length of task n.
C_n	Processing density of task n.
t_n^L	Time for local processing in IoT device n.
E_n^L	Energy consumption for processing task in IoT device n.
k_L	A coefficient related to power in IoT device.
p_n^L	Power consumption for IoT device n.
p_n	Transmission power of IoT device n.
a_n	Offloading decision of IoT device n.
b_n	Connectivity between IoT device n and fog node.
f_m	Processing capability of fog node m.
f_n^L	Processing capability of IoT device n.
α_m	Cost for processing one bit.
β_m	Discount coefficient.
R_n^m	Tranmission rate between IoT user n and fog m.
t_n^{tr}	Transmission time of task n.
E_n^{tr}	Transmission energy of task n.
t_n^m	Execution time of task n in fog node m.
c	Fees charged by fog nodes.
A_{-n}	Offloading decisions for all IoT devices except IoT device n.

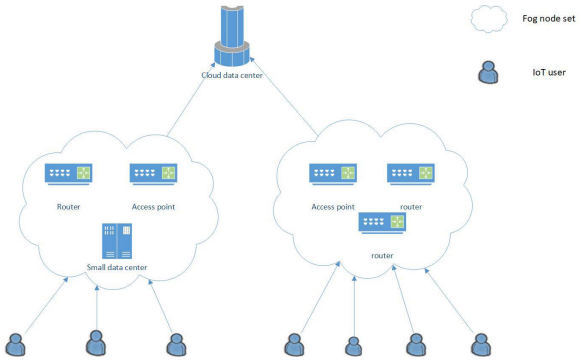

FIGURE 1. System model.

A. HIERARCHICAL STRUCTURE

Denote the set of N IoT devices and the set of M fog nodes by $N = \{1,2,\ldots\ldots,N\}$ and $M = \{1,2,\ldots\ldots,M\}$, respectively. IoT devices can process tasks locally or offload to the second-tier fog node and the third-tier cloud data center. If the task is delay-insensitive, we can offload it to a cloud data center. Tasks are processed in the cloud data center higher than the fog nodes. Cloud data centers cannot meet the needs of delay-sensitive tasks. So we offload the delay-sensitive tasks to fog nodes. We use two items to describe the computation task of IoT device n ($n \in N$), $T_n = \{L_n, C_n\}$, where L_n represents the length of T_n and C_n represents the processing density (in CPU cycles/bit) of T_n. Define the association vector of IoT device n as $a_n = \{a_{n,0}, a_{n,1}, \ldots\ldots, a_{n,M}\}$, where $a_{n,x} \in \{0, 1\}$, $x \in \{0\} \bigcup M$, with $\sum_{x\in\{0\}\bigcup M} a_{n,x} = 1$. a_n denotes offloading decision. If $a_{n,x}$ is 1, the task is processed in device x. $a_{n,0}$ means the task is processed locally. We further define $A = \{a_1^T, a_2^T, \ldots, a_n^T\}^T$. A includes the decision of all IoT devices. Since fog nodes are heterogeneous, not all fog nodes can be used to process tasks. Define another vector $b_n = \{b_{n,1}, bn, 2\ldots\ldots, bn, M\}$. b_n represents the connectivity between IoT device n and fog nodes. If $b_{n,i}$ is 1, IoT device n can offload tasks to fog i.

B. TASK PROCESSED LOCALLY

if $a_{n,0} = 1$, we processed the task locally. Let f_n^L denotes the processing capability (i.e., the amount of CPU frequency in CPU cycles/s) at IoT device n. The power consumption for

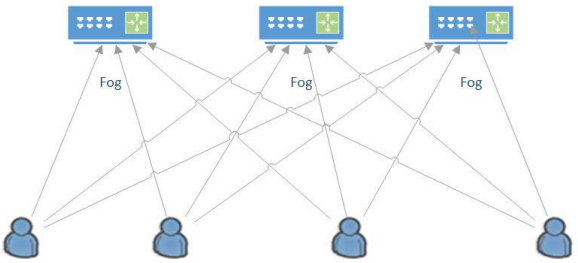

FIGURE 2. Communication between fog and IoT layer.

IoT device n is expressed as follows [26]–[28]:

$$p_n^L = k_L \left(f_n^L\right)^3 \tag{1}$$

where k_L is a coefficient related to the power of IoT device n. The time for local processing in IoT device n as follows:

$$t_n^L = \frac{L_n C_n}{f_n^L} \tag{2}$$

Accordingly, the energy consumption of IoT device n for local processing is expressed as:

$$E_n^L = p_n^L t_n^L = k_L L_n C_n \left(f_n^L\right)^2 \tag{3}$$

C. TASK OFFLOADED TO FOG

As shown in Figure 2, the IoT device can offload tasks to the fog nodes in the area. There is more than one such fog node, how should we choose. When $\sum_{x\in M} a_{n,x} = 1$, the task is offloaded to fog. We assume the task is offloaded to fog m. The offloaded time of the task includes two-part, one is the transmission time and another is processing time. In this article, we do not consider the time of data return, the processing result is generally very small. Accordingly, when transmitting task T_n, the transmission time is expressed as:

$$t_n^{tr} = \frac{L_n}{R_n^m} \tag{4}$$

where R_n^m represents the transmission rate between IoT user n and fog m [24], [25]. Let p_n denotes the transmission power. And the energy consumption for transmitting task is expressed as:

$$E_n^{tr} = p_n t_n^{tr} = \frac{p_n L_n}{R_n^m} \tag{5}$$

Let f_m denotes the processing capability of fog node m. The fog node will evenly allocate CPU resources for each task. The execution time of t_n^m is expressed as:

$$t_n^m = \frac{L_n C_n \sum_{i=1}^{N} a_{i,m}}{f_m} \tag{6}$$

Offloading tasks to the fog node can save energy, but because the operation and calculation of the fog node require costs, the fog node will charge a certain amount of waste from the IoT device. To motivate more IoT devices to offload tasks, we define the cost of processing a bit in fog node m is expressed as:

$$c = L_n \left(\alpha_m - \beta_m \sum_{i=1}^{N} a_{i,m}\right) \tag{7}$$

α_m and β_m is set by fog node m. α_m represents the cost for processing one bit. β_m is the discount coefficient.

D. PROBLEM STATEMENT

Due to the selfish nature of the device, each device maximizes its interests as much as possible. We define the utility function of IoT device n as:

$$U_n(a_n, A_{-n}) = a_{n,0} * (\lambda^E E_n^L + \lambda^T t_n^L) + \sum_{m=1}^{M} a_{n,m}(\lambda^E E_n^{tr} + \lambda^T (t_n^{tr} + t_n^m) + \lambda^C c) \tag{8}$$

where A_{-n} is decision of all IoT devices except n. The goal of the target IoT device is expressed as:

$$\min_{a_n} U_n(a_n, A_{-n}) \tag{9a}$$

$$\text{s.t.} \sum_{m\in\{0\}\bigcup M} a_{n,m} = 1 \tag{9b}$$

$$a_{n,m} \le b_{n,m} \quad \forall m \in M \tag{9c}$$

$$a_{n,m} \in \{0, 1\} \quad \forall m \in M \tag{9d}$$

(9a) ensures that the task can only be offloaded to at most one fog node. (9b) indicates that the selected fog node should be connected to the IoT device.

IV. COMPUTATION OFFLOADING

We formally define game $G = \{N, A, U\}$. which consists of three parts:

- N is the set of players and N represents IoT devices in this article
- A is the set of decision space of all IoT devices.
- U is the set of the utility of all players, U_n is the utility of IoT device n.

Each IoT device will maximize its interests according to (9), which will eventually reach a Nash Equilibrium. A_{-n} denotes the offloading decisions for all IoT devices except n.

Definition 1 (Nash Equilibrium): For game G, we call $A^* = \left(a_1^*, a_2^*, \ldots\ldots, a_n^*\right)$ Nash Equilibrium if and only if no IoT devices can further improve its utility by changing its decision at the equilibrium point A^*.

$$u_n(a_n^*, A_{-n}^*) \le u_n(a_n, A_{-n}^*) \quad \forall a_n \in A \; \forall n \in N \tag{10}$$

A. NASH EQUILIBRIUM EXISTENCE

Definition 2 (Weighted Potential Game): Define a $W = (w_1, w_2, \ldots\ldots, wn)$ denotes a vector of positive numbers. A game is called a weighted potential game if it admits a w-potential function P such that for every player $n \in N$ and offloading vectors $a_n, a_n' \in A$

$$U_n(a_n, A_{-n}) - U_n(a_n', A_{-n}) = w_n(P(a_n, A_{-n}) - P(a_n', A_{-n})) \tag{11}$$

The user scheduling game G possesses at least one NE when game G is a weighted potential game. Since the weighted potential game has the finite improvement property (FIP) that any better response updating process must be finite and lead to a Nash Equilibrium. So we first prove G is a potential game.

$$\begin{aligned}
&U_n(a_n, A_{-n}) \\
&\quad = a_{n,0} * (\lambda^E E_n^L + \lambda^T t_n^L) \\
&\qquad + \sum_{m=1}^{M} a_{n,m}(\lambda^E E_n^{tr} + \lambda^T (t_n^{tr} + t_n^m) + \lambda^C c) \\
&\quad = a_{n,0} * (\lambda^E k_L L_n C_n \left(f_n^L\right)^2 + \lambda^T \frac{L_n C_n}{f_n^L}) \\
&\qquad + \sum_{m=1}^{M} a_{n,m}(\lambda^T (\frac{L_n}{R_n^m} + \frac{L_n C_n \sum_{i=1}^{N} a_{i,m}}{f_m}) \\
&\qquad + \lambda^E \frac{p_n L_n}{R_n^m} + \lambda^c L_n \left(\alpha_m - \beta_m \sum_{i=1}^{N} a_{i,m}\right)) \qquad (12) \\
&U_n(a_n, A_{-n}) - U_n(a_n', A_{-n}) \\
&\quad = (a_{n,0} - a_{n,0}')(\lambda^E k_L L_n C_n \\
&\qquad \left(f_n^L\right)^2 + \lambda^T \frac{L_n C_n}{f_n^L}) + \sum_{m=1}^{M} (a_{n,m} - a_{n,m}')(\lambda^T (\frac{L_n}{R_n^m} \\
&\qquad + \frac{L_n C_n \sum_{i=1}^{N} a_{i,m}}{f_m}) + \lambda^E \frac{p_n L_n}{R_n^m} \\
&\qquad + \lambda^c L_n \left(\alpha_m - \beta_m \sum_{i=1}^{N} a_{i,m}\right)) \qquad (13)
\end{aligned}$$

We first define the function $Q(A)$ as the weighted aggregate utility of all users

$$Q(a_n, A_{-n}) = \sum_{n=1}^{N} \frac{1}{\lambda^T} U_n(a_n, A_{-n}) \qquad (14)$$

We further define the function $Q(A)'$as the weighted aggregate utility of all users if each user is alone in the game

$$Q(a_n, A_{-n})' = \sum_{n=1}^{N} \frac{1}{\lambda^T} U_n(a_n, 0) \qquad (15)$$

0 represents no other players. We define a function $P(A)$.

$$P(a_n, A_{-n}) = \frac{Q(a_n, A_{-n}) + Q(a_n, A_{-n})'}{2} \qquad (16)$$

We use the function defined in (16) and $w_n = \frac{1}{\lambda_T}$, the proof process is expressed as follows.

$$\begin{aligned}
&P(a_n, A_{-n}) - P(a_n', A_{-n}) \\
&= \frac{Q(a_n, A_{-n}) - Q(a_n', A_{-n})}{2} \\
&\quad + \frac{Q(a_n, A_{-n})' - Q(a_n', A_{-n})'}{2} = \sum_{n=1}^{N} \frac{1}{2\lambda^T} (U_n(a_n, A_{-n}) \\
&\quad - U_n(a_n', A_{-n})) + \sum_{n=1}^{N} \frac{1}{2\lambda^T} (U_n(a_n, 0) - U_n(a_n', 0)) \\
&= \frac{1}{\lambda^T}(a_{n,0} - a_{n,0}')(\lambda^E k_L L_n C_n \left(f_n^L\right)^2 + \lambda^T \frac{L_n C_n}{f_n^L}) \\
&\quad + \sum_{m=1}^{M} \frac{1}{\lambda^T}(a_{n,m} - a_{n,m}')(\lambda^T (\frac{L_n}{R_n^m} + \frac{L_n C_n}{f_m}) + \lambda^E \frac{p_n L_n}{R_n^m} \\
&\quad + \lambda^c (L_n(\alpha_m - \beta_m))) \\
&\quad + \sum_{k=1, k\neq n}^{N} \sum_{m=1}^{M} \frac{1}{2}(\alpha_{n,m} - \alpha_{n,m}')\alpha_{k,m} \frac{L_n C_n}{f_m} \\
&\quad + \sum_{k=1, k\neq n}^{N} \sum_{m=1}^{M} \frac{1}{2}\alpha_{k,m}'(\alpha_{n,m} - \alpha_{n,m}') \frac{L_n C_n}{f_m} \\
&\quad - \sum_{k=1, k\neq n}^{N} \sum_{m=1}^{M} \frac{\lambda^c}{2\lambda^T}(\alpha_{n,m} - \alpha_{n,m}')\alpha_{k,m}\beta_m \\
&\quad - \sum_{k=1, k\neq n}^{N} \sum_{m=1}^{M} \frac{\lambda^c}{2\lambda^T}\alpha_{k,m}'(\alpha_{n,m} - \alpha_{n,m}')\beta_m \\
&= \frac{1}{\lambda^T}(a_{n,0} - a_{n,0}')(\lambda^E k_L L_n C_n \\
&\quad \times \left(f_n^L\right)^2 + \lambda^T \frac{L_n C_n}{f_n^L}) + \sum_{m=1}^{M} (a_{n,m} - a_{n,m}')(\lambda^T (\frac{L_n}{R_n^m} \\
&\quad + \frac{L_n C_n \sum_{i=1}^{N} a_{i,m}}{f_m}) + \lambda^E \frac{p_n L_n}{R_n^m} \\
&\quad + \lambda^c L_n \left(\alpha_m - \beta_m \sum_{i=1}^{N} a_{i,m}\right)) \\
&= \frac{1}{\lambda^T}(U_n(a_n, A_{-n})) - U_n(a_n', A_{-n}))
\end{aligned}$$

In conclusion, the user scheduling game G is a weighted potential game with the potential function as given in (16). There exists at least one NE point. At the Nash Equilibrium point, no IoT user would change its decision thus no update message would be broadcasted.

B. DISTRIBUTED ALGORITHM FOR TASK OFFLOADING

The centralized algorithms are generally time-complex and require a management center to supervise them. Once the central server crashes, the entire program will not run. So this article uses a distributed task offloading method. So this article uses a distributed task offloading method. Algorithm pseudo-code shows in Algorithm 1. From algorithm 1, we can get the time complexity of the algorithm is proportional to the number of competing IoT devices. And we prove the offloading game G is a weighted potential game. FIP is a feature of the potential function. FIP shows that the algorithm can reach a Nash equilibrium in a finite number of steps [25], [26], [29]. Therefore, our algorithm has a time complexity of $O(SN)$. S represents the number of steps the IoT device takes to reach the Nash equilibrium.

If we know a better decision A at the beginning of the algorithm, the convergence speed of the algorithm will increase a lot. We can use UCB, Thompson sampling, and other methods to determine the initial decision. In some cases,

Algorithm 1: Distributed Task Offloading Algorithm

Input: Information about fog nodes(ie:f_n)
Output: Offloading decisions of all IoT devices
Initialize $a_n = (1, 0, \ldots\ldots, 0)$ for all IoT devices;
$A = (a_1, a_2, \ldots\ldots, a_n)$;
while *A is not changed* **do**
  All IoT device parallel compute the best decision according(9); $a_n^* = \arg\min_{a_n} U_n(a_n, A_{-n})$;
  Compete for the decision updating opportunity;
  if *win the competition opportunity* **then**
    $a_n = a_n^*$;
    broadcast update message;
  else
    a_n keeps unchanged;

we can appropriately reduce the performance in exchange for the algorithm to converge quickly. So we can use Algorithm 2. ε means acceptable utility loss. ε is customized by the user. Algorithm 1 may get better Utility, but Algorithm 2 can get faster convergence speed. We can choose an algorithm according to the actual situation.

Algorithm 2: ε Distributed Task Offloading Algorithm

Input: Information about fog nodes(ie:f_n)
Output: Offloading decisions of all IoT devices
Initialize a_n for all IoT devices according to Thompson sampling;
$A = (a_1, a_2, \ldots\ldots, a_n)$;
while *A is not changed* **do**
  All IoT device will parallel compute the best decision according(9);
  $a_n^* = \arg\min_{a_n} U_n(a_n, A_{-n})$;
  if $U_n(a_n, A_{-n}) - U_n(a_n', A_{-n}) > \varepsilon$ **then**
    Compete for the decision updating opportunity;
  if *win the competition opportunity* **then**
    $a_n = a_n^*$;
    broadcast update message;
  else
    a_n keeps unchanged;

C. FAIRNESS BETWEEN FOG NODES

If most IoT devices in the network select the same fog node for computing offloading, the energy of the fog node will be exhausted quickly. And node failure can sometimes cause partial paralysis of the network, so based on this, we consider the balance of offloading. We will offload tasks to nodes with sufficient power as much as possible. But it is more complicated for IoT devices to monitor the power of fog devices. So we let the fog nodes check their battery periodically and adjust the processing rate. See Algorithm 3 for a specific method. The cycle time can be set by the fog

Algorithm 3: Calculation Rate Adjustment

a =total battery;
for *End of a cycle* **do**
  b =remaining battery;
  if *Electricity is less than 50 percent* **then**
    $f_n = \frac{b}{a} f_n$
  else
    f_n keeps unchanged;

providers. The degree of imbalance shows the imbalance among the fog nodes and is expressed as follows [20]:

$$DI = \frac{\arg\max_m \sum_{i=1}^{N} a_{i,m} L_n - \arg\min_m \sum_{i=1}^{N} a_{i,m} L_n}{\frac{\sum_{i=1}^{N} L_n}{m}} \tag{17}$$

Formula (18) can only reflect the difference between the node with the largest processing capacity and the node with the smallest processing capacity, but it cannot reflect the status of all nodes. So we define the standard deviation of the load imbalance.

$$V = \sqrt{\frac{\sum_{m=1}^{M} (\sum_{i=1}^{N} L_n a_{i,m} - \frac{\sum_{i=1}^{N} L_n}{m})^2}{M}} \tag{18}$$

The larger the DI and V, the more unfair of offloading. Otherwise, the offloading algorithm has good fairness. In some scenarios, some nodes will be more important. So that we can adjust the 50 percent in Algorithm 3.

V. PERFORMANCE EVALUATION

We set up 3 fog nodes and 20 IoT devices in the experiment. The processing density of the fog node is evenly distributed in [100000, 200000], and the processing density of the IoT device is evenly distributed in [10000, 20000]. Length of tasks generated by IoT devices l_n and c_n are distributed in [1000, 2000][24]–[26]. Set λ_T, λ_E,and λ_C to 0.4, 0.2, 0.2 respectively. λ_T, λ_E,and λ_C can be set according to the actual situation. If we pay more attention to processing time, we can set λ_T larger. But we have to make sure that $\lambda_T + \lambda_E + \lambda_C = 1$. The transmission power of the IoT device is set to 0.02. Assume that the transmission rate between all nodes is the same, which is 100. ε is evenly distributed in [0, 1]. ε can be adjusted according to your needs. ε is usually not very large and a large value will result in low utility. The software environment we utilize is Python 3.7 on windows.

See from FIGURE 3, We compare the proposed algorithm with LCO(Local computation Only) and RCO(Random computation offload) algorithms, and the DTO algorithm can achieve the smallest cost. Followed by the *ε-DTO* algorithm. Although the *ε-DTO* algorithm does not achieve the smallest utility, its convergence speed is faster than the DTO algorithm. The LRO algorithm, because it chooses to process locally every time, so the processing speed remains

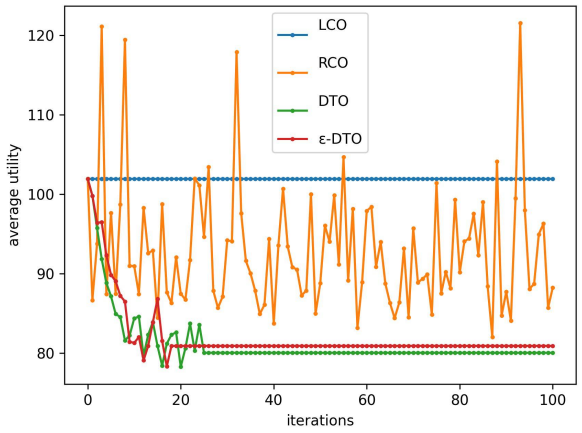

FIGURE 3. Average utility in different iterations.

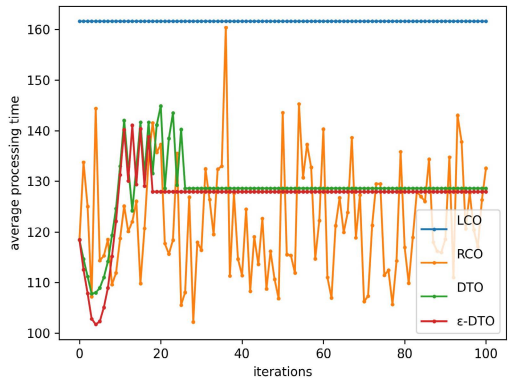

FIGURE 4. Average processing time in different iterations.

unchanged. The randomness of the RCO algorithm is too large, and occasionally it can achieve good results, but it is unstable.

FIGURE 4 shows the processing time of different iterations. We can see that the average task processing time is the same as the overall utility trend in FIGURE 3. Because the RCO algorithm selects the node to offload randomly, the effect is very unstable. The two algorithms we proposed can achieve an almost similar effect. In terms of processing time alone, our proposed algorithm is also superior to other algorithms. FIGURE 5 shows the average energy consumption in different iterations. We can see from FIGURE 5 that our proposed algorithm is superior to other algorithms. *ε-DTO* converges faster but doesn't get better energy consumption. *DTO* converges slower but the effect is better. In different situations, we can choose different algorithms.

FIGURE 6 shows the average utility value under different numbers of IoT devices. Run 1000 rounds for each number of nodes to average the utility value. As the number of IoT increases, the utility will increase due to the heavier load of the fog node. Overall showing an upward trend. When the number of IoT is small, RCO can achieve better results. However, once the number of IoT devices increases, the RCO

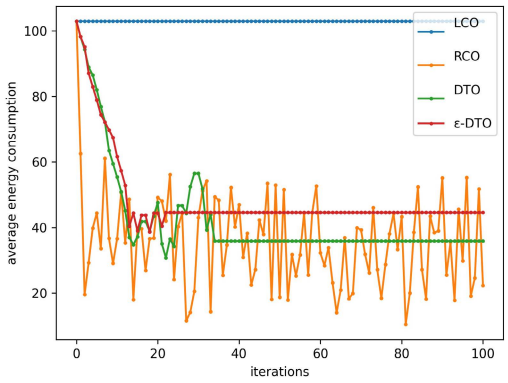

FIGURE 5. Average energy consumption in different iterations.

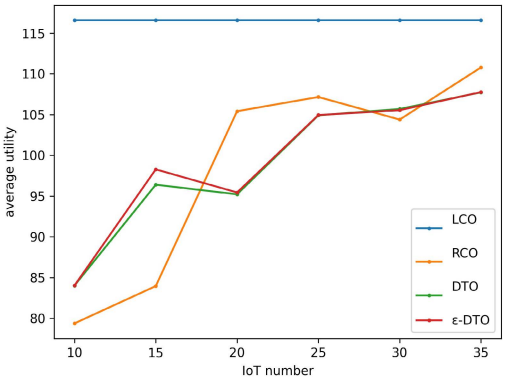

FIGURE 6. Average utility with different IoT numbers.

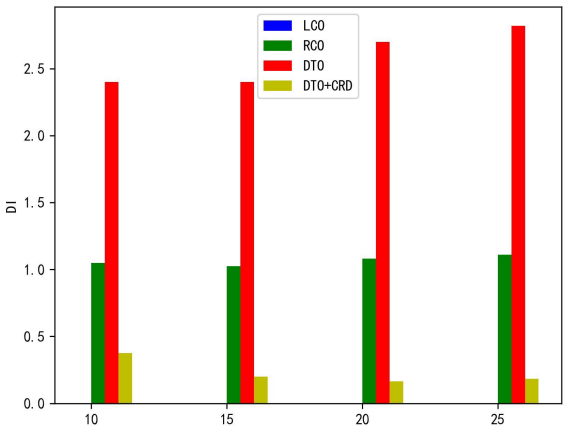

FIGURE 7. DI with different IoT numbers.

algorithm may cause multiple IoT devices to compete for the same fog node, increasing the utility function. And our proposed algorithm is more stable and will not lead to a high failure rate.

DI reflects the difference between the node with the largest processing capacity and the node with the smallest processing capacity. FIGURE 7 shows DI in different IoT numbers. Since the LCO algorithm always chooses local processing, the value of DI is always 0. It can be seen that the DI with

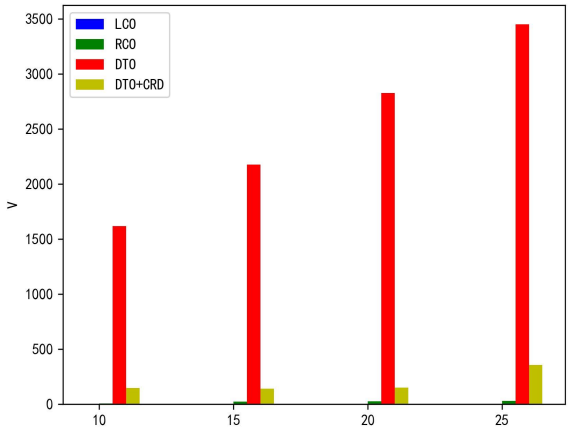

FIGURE 8. V with different IoT numbers.

a large DTO is higher than the LCO and RCO. This shows that although the DTO algorithm reduces utility, it makes the network unfair, which leads to premature failure of some nodes. To solve this problem, we propose the CRD algorithm, the combination of DTO and CRD algorithm, which effectively reduces the DI and improves the fairness of the algorithm.

V reflect the status of all node. It can be seen from FIGURE 8 that the V of RCO is the smallest. This is because when there are enough tasks, the RCO algorithm selects each node more averagely. The RCO algorithm is fair in the long run, but does not achieve good performance. Compared with the other algorithm, the algorithm combining DTO and CRD greatly increases fairness while ensuring the consistency of utility.

VI. CONCLUSION

This article takes the process of IoT device competition for fog nodes as a game. And uses the finite improvement property of the Weighted Potential Game to prove the Nash equilibrium. And this article considers offloading fairness, which can delay the life cycle of the network, and will not allow a single fog node to process too many tasks and run out of energy. Based on the game, this paper design Distributed task offloading algorithm(DTO) and ε Distributed task offloading algorithm(ε-DTO). The computation rate adjustment(CRD) algorithm further improves the balance of offloading. The experiment proves that the proposed Distributed task offloading algorithm can get the best utility. ε Distributed task offloading can get a good utility and better convergence speed. And DTO combines with CRD greatly improves the fairness of the original algorithm.

We can consider more detailed information(channel selection, power control, etc) to minimize utility in future work. We can use methods such as deep learning[34], [35] to determine a better initial offload decision. We can also offload tasks to multiple fog nodes for parallel processing. This involves the issue of offloading orders. And we can study how to put services in the right place to minimize the objective function[23], [36], [37].

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[2] M. Chiang, S. Ha, C.-L. I, F. Risso, and T. Zhang, "Clarifying fog computing and networking: 10 questions and answers," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 18–20, Apr. 2017.

[3] E. Ahmed, A. Naveed, A. Gani, S. H. A. Hamid, M. Imran, and M. Guizani, "Process state synchronization for mobility support in mobile cloud computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[4] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *J. Netw. Comput. Appl.*, vol. 59, pp. 46–54, Jan. 2016.

[5] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[6] Cisco. (2015). *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*. [Online]. Available: http://www.cisco.com/c/dam/enus/solutions/trends/iot/docs/computingoverview.pdf

[7] H. F. Atlam, R. J. Walters, and G. B. Wills, "Fog computing and the Internet of Things: A review," *Big Data Cogn. Comput.*, vol. 2, no. 2, p. 10, 2018.

[8] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 3909–3914.

[9] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S., "The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1728–1739, May 2016.

[10] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "Internet of Things (IoT): A vision, architectural elements, and future directions," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2586–2595, Nov. 2017.

[11] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang, "Selective offloading in mobile edge computing for the green Internet of Things," *IEEE Netw.*, vol. 32, no. 1, pp. 54–60, Jan./Feb. 2018.

[12] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 34–40, Apr. 2017.

[13] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart., 2018.

[14] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no. 1, pp. 124–143, 1996.

[15] S. Luo, X. Chen, Z. Zhou, Chen, X., and W. Wu, "Incentive-aware micro computing cluster formation for cooperative fog computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2643–2657, May 2020.

[16] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M.-T. Zhou, "FEMTO: Fair and energy-minimized task offloading for fog-enabled IoT networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4388–4400, Jun. 2019.

[17] S. J. Darak and M. K. Hanawal, "Multi-player multi-armed bandits for stable allocation in heterogeneous ad-hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2350–2363, Oct. 2019.

[18] M. Abedi and M. Pourkiani, "Resource allocation in combined fog-cloud scenarios by using artificial intelligence," in *Proc. 5th Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Apr. 2020, pp. 218–222.

[19] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

[20] M. K. Hussein and M. H. Mousa, "Efficient task offloading for IoT-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37191–37201, 2020.

[21] D. Xu, Q. Li, and H. Zhu, "Energy-saving computation offloading by joint data compression and resource allocation for mobile-edge computing," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 704–707, Apr. 2019.

[22] G. Yang, L. Hou, X. He, D. He, S. Chan, and M. Guizani, "Offloading time optimization via Markov decision process in mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2483–2493, Feb. 2021.

[23] T. Huang, W. Lin, C. Xiong, R. Pan, and J. Huang, "An ant colony optimization-based multiobjective service replicas placement strategy for fog computing," *IEEE Trans. Cybern.*, vol. 51, no. 11, pp. 5595–5608, Nov. 2021.

[24] H. Tan and L. Zhu, "Overall computing offloading strategy based on deep reinforcement learning in vehicle fog computing," *J. Eng.*, vol. 2020, no. 11, pp. 1080–1087, Nov. 2020.
[25] Z. Liu, Y. Yang, Y. Chen, K. Li, Z. Li, and X. Luo, "A multi-tier cost model for effective user scheduling in fog computing networks," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Apr. 2019, pp. 1–6.
[26] Q. Luo, C. Li, T. H. Luan, W. Shi, and W. Wu, "Self-learning based computation offloading for internet of vehicles: Model and algorithm," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 5913–5925, Sep. 2021.
[27] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.
[28] G. Qu, "What is the limit of energy saving by dynamic voltage scaling?" in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design*, 2001, pp. 560–563.
[29] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for IoT systems: A computation offloading game," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3246–3257, Aug. 2018.
[30] V. Valls, G. Iosifidis, and T. Salonidis, "Maximum lifetime analytics in IoT networks," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2019, pp. 1369–1377.
[31] X. Lyu, C. Ren, W. Ni, H. Tian, and R. P. Liu, "Distributed optimization of collaborative regions in large-scale inhomogeneous fog computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 574–586, Mar. 2018.
[32] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, and G. B. and Giannakis, "Distributed online optimization of fog computing for selfish devices with out-of-date information," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, pp. 7704–7717, Nov. 2018.
[33] Y. Xiao and M. Krunz, "Distributed optimization for energy-efficient fog computing in the tactile internet," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2390–2400, Nov. 2018.
[34] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.
[35] E. Baccarelli, M. Scarpiniti, A. Momenzadeh, and S. S. Ahrabi, "Learning-in-the-fog (LiFo): Deep learning meets fog computing for the minimum-energy distributed early-exit of inference in delay-critical IoT realms," *IEEE Access*, vol. 9, pp. 25716–25757, 2021.
[36] M. S. Raghavendra, P. Chawla, and A. Rana, "A survey of optimization algorithms for fog computing service placement," in *Proc. 8th Int. Conf. Rel., Inf. Technol. Optim.*, Jun. 2020, pp. 259–262.
[37] H. Sami, A. Mourad, H. Otrok, and J. Bentahar, "Demand-driven deep reinforcement learning for scalable fog and service placement," *IEEE Trans. Services Comput.*, early access, Apr. 27, 2021, doi: 10.1109/TSC.2021.3075988.

SUN YU-JIE received the bachelor's degree in network engineering from Wuhan Textile University, Wuhan, China, in 2017. She is currently pursuing the master's degree in computer science and technology with Zhejiang Normal University, Jinhua, China. Since 2018, she has been working as an Assistant Software Engineer in Wuhan. Her research interests include the Internet of Things, cloud computing, and fog computing.

WANG HUI received the Ph.D. degree in electronic engineering from Shanghai Jiao Tong University, China, in 2008. From 2008 to 2009, he was a Postdoctoral Fellow with the Department of Computer Science, Evry University, France. From 2009 to 2011, he was a BK21 Postdoctoral Fellow at the Department of IT Convergence, POSTECH, South Korea. He is currently an Associate Professor with the Department of Computer Science, Zhejiang Normal University. His research interests include compressive sensing, edge/fog computing, distributed optimization, online optimization, edge intelligence, and self-adaptive IoT.

ZHANG CHENG-XIANG received the bachelor's degree in software engineering from Zhejiang Normal University, in 2016. He is currently a Graduate Student majoring in electronic information engineering with Zhejiang Normal University. He is engaged in the research of the Internet of Things. He is studying the research project of fog and edge computing. His research interests include operating systems, cloud computing, and fog computing.