Workload Balancing in Mobile Edge Computing for Internet of Things: A Population Game Approach

Dongqing Liu*†, Abdelhakim Hafid*, Lyes Khoukhi†
*Department of Computer Science and Operational Research
University of Montreal, QC, Canada
Email: ahafid@iro.umontreal.ca
†Environment and Autonomous Networks Lab (ERA)
University of Technology of Troyes, France
Email: {dongqing.liu, lyes.khoukhi}@utt.fr

***Abstract*—Mobile edge computing (MEC) is an emerging paradigm that provides radio access networks with augmented resources to meet the requirements of Internet of Things (IoT) services. MEC allows IoT devices to offload delay sensitive and computation intensive tasks to edge clouds deployed at base stations (BSs). Offloading tasks to edge clouds can alleviate the computing and battery limitations of IoT devices. However, task offloading in MEC for IoT may face serious transmission latency and computation latency problems with massive number of IoT devices. Moreover, some edge clouds can be overloaded due to the spatially inhomogeneous distributions of IoT tasks. To solve these problems, we investigate the workload balancing problems to minimize the transmission latency and computation latency in task offloading process while considering the limited bandwidth resources of BSs and computation resources in edge clouds. We formulate the workload balancing problem as a population game in order to analyze the aggregate offloading decisions. We analyze the aggregate offloading decisions of mobile users through evolutionary game dynamics and show that the game always achieves a Nash equilibrium (NE). We further propose two workload balancing algorithms based on evolutionary dynamics and revision protocols. Simulation results show that our proposed workload balancing algorithms can achieve better performance than existing solutions.**

***Index Terms*—Task offloading, Population Game, Mobile edge computing, Internet of Things.**

I. Introduction

IoT is proposed to equip everyday objects with electronics, software, sensors, and network connectivity, and bring the vision of a connected world into reality [1]. However, computation-intensive applications, such as e-health, automatic driving, and industrial automation, consume large amounts of computing and storage capabilities of IoT devices. These sophisticated applications have stringent requirements of computation resources and processing delay on IoT Devices (IoTDs). However, IoTDs are resource-constrained and have limited computational capacities and battery life. Running computation intensive applications on IoTDs would result in high energy consumption and long processing delay [2]. The conflict between computation intensive applications and resource constrained IoTDs brings a significant challenge for future mobile development. MEC is envisioned to be a promising solution to address this challenge, with the objective to provide cloud computing capabilities to IoTDs through radio access network [3]. By offloading computation intensive tasks to edge cloud (or MEC server) in proximity, the local energy consumption on IoTDs can be reduced and the local processing delay may be shortened [4].

To offload computation intensive tasks to edge cloud, task-related data should be transferred between IoTDs and edge cloud through base station (BS). If BS is congested by large amounts of IoTDs choosing to offload tasks simultaneously, the quality of experience and quality of service of IoTDs will not be guaranteed [5, 6]. Moreover, facing the rapid increase of IoTDs and massive offloading tasks, the resource bottleneck of edge cloud becomes significant, since edge cloud has relatively limited resources compared to cloud computing [7]. Thus, lack of proper offloading coordination among large amounts of self-interested IoTDs may lead to serve interferences in wireless transmission and load unbalance in edge clouds. [8, 9]. As a result, designing an energy-efficient offloading mechanism while satisfying the processing delay requirements becomes a challenging problem, especially when large amounts of IoTDs compete for limited resources.

In this paper, we propose a population game based approach to investigate workload balancing problem for MEC in the context of IoT. Population game is envisioned as a powerful tool to model strategic interactions among large amounts of agents [10, 11]. Specifically, we model the offloading decision making problem among large amounts of competing IoTDs as a population game, wherein IoTDs are self-interested agents that make offloading decisions individually. The main contributions of this paper are summarized as follows:

- *Population game model formulation:* We formulate MEC workload balancing problem as a population game and propose an IoT Device classification model. We design an inference affected queueing model that can capture the inference among IoTDs. We use $\alpha-$ utility function to implement different kinds of workload balancing.
- *Evolutionary game dynamics analysis:* We calculate Nash Equilibrium (NE) dynamically, i.e., IoTDs can change their offloading decisions through some learning mechanism. The learning mechanism is defined as a revision protocol that allows IoTDs to adjust their offloading decisions based on decisions of other IoTDs in proximity.

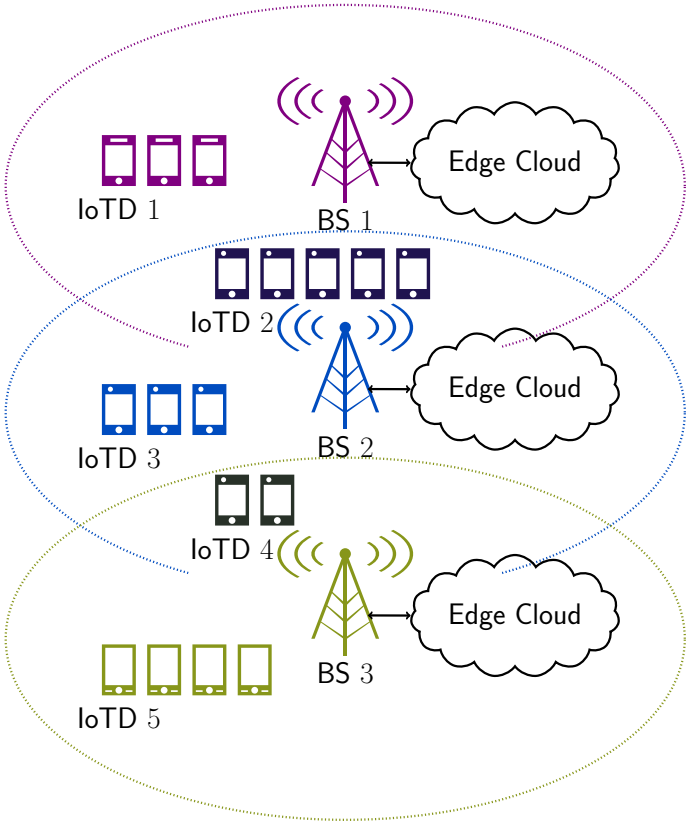

Fig. 1. MEC workload balancing model. n IoTDs offload computation intensive tasks to m edge clouds by BSs. The available offloading strategies depend on the location of IoTDs, e.g., IoTD1 can only offload tasks to BS1, while IoTD2 can offload tasks to BS1 or BS2, since IoTD can only access BSs in proximity.

The evolutionary process of IoTDs' offloading strategies can be modeled by evolutionary game dynamics (i.e., a differential equation). The evolutionary game dynamics describes the variation of IoTDs' offloading decisions until an NE is obtained.

- *Workload balancing algorithms:* We propose two workload balancing algorithms, namely centralized workload balancing algorithm and decentralized workload balancing algorithm, based on the concept of evolutionary dynamics and revision protocols, respectively. We show that these algorithms can achieve an NE. Simulation results illustrate the evolutionary dynamics and show that the proposed algorithms can achieve efficient workload balancing in BSs and edge clouds.

The remainder of this paper is organized as follows. Section II introduces the system model of MEC workload balancing. Section III proposes $\alpha-$utility function based workload balancing model. Section IV proposes two population game based workload balancing algorithms. Section V shows the evolutionary dynamics of three revision protocols and evaluate the performance of our proposed algorithms. Section VI concludes the paper.

TABLE I
BASIC NOTATION

Elements	
$\mathbf{B}_m$	Base Station m
$\mathbf{E}_m$	Edge Cloud m, collocated with $\mathbf{B}_m$
$\mathbf{D}_q$	Internet of Thing Device q
$\mathbf{C}_n$	Class n, Set of Internet of Thing Devices
$\mathbf{D}_n^*$	The IoTD with minimum data size in Class n
$\mathbf{D}_n^m$	An IoTD in Class n choosing $\mathbf{B}_m$
Sets	
$\mathbb{B}=\{\mathbf{B}_m\}_{m\in\mathcal{M}}$	Set of Base Stations, $\lvert\mathcal{M}\rvert=M$
$\mathbb{D}=\{\mathbf{D}_q\}_{q\in\mathcal{Q}}$	Set of Internet of Thing Devices, $\lvert\mathcal{Q}\rvert=Q$
$\mathbb{C}=\{\mathbf{C}_n\}_{n\in\mathcal{N}}$	Set of Classes or Partitions of $\mathbb{D}$, $\lvert\mathcal{N}\rvert=N$
$\mathbb{B}^n=\{\mathbf{B}_m\}_{m\in\mathcal{M}^n}$	Subset of Base Stations that are available for $\mathbf{C}_n$
Parameters	
$L_q\in\mathbb{R}^2$	The location of $\mathbf{D}_q$
B_q	The length of data flow from $\mathbf{D}_q$
E_q	The length of computation flow from $\mathbf{D}_q$
λ_q	Task generation rate from $\mathbf{D}_q$
Z^n	The number of IoTDs in $\mathbf{C}_n$, $\lvert\mathbf{C}_n\rvert=Z^n$
$\hat{Z}^n$	The number of IoTDs in $\mathbf{C}_n$ after replacement
C^n	Computational density per unit size of data in $\mathbf{C}_n$
$L^n\in\mathbb{R}^2$	The location of $\mathbf{C}_n$
B^n	The length of data flow from $\mathbf{D}_n^*$
E^n	The length of computation flow from $\mathbf{D}_n^*$
λ^n	Task generation rate from $\mathbf{D}_n^*$
θ^n	Traffic generation density from $\mathbf{D}_n^*$
η^n	Computation generation density from $\mathbf{D}_n^*$
Variables	
a_m^n	The number of IoTDs in $\mathbf{C}_n$ choosing $\mathbf{B}_m$
$\boldsymbol{a}^n$	Class state vector $\boldsymbol{a}^n=[a_m^n]_{m\in\mathcal{M}^n}$
$\boldsymbol{a}$	Population state vector $\boldsymbol{a}=[\boldsymbol{a}^n]_{n\in\mathcal{N}}$

II. System Model

We consider a cellular network consisting of a set of BSs, denoted by $\mathbb{B}=\{\mathbf{B}_m\}_{m\in\mathcal{M}}$, where $\mathcal{M}=\{1,2,\cdots,M\}$. We denote IoTDs within the coverage area of these BSs by a set $\mathbb{D}=\{\mathbf{D}_q\}_{q\in\mathcal{Q}}$, where $\mathcal{Q}=\{1,2,\cdots,Q\}$. A partition of set $\mathbb{D}$ is denoted by $\mathbb{C}=\{\mathbf{C}_n\}_{n\in\mathcal{N}}$, where $\mathcal{N}=\{1,2,\cdots,N\}$. $\mathbb{D}$ can be partitioned into N subsets; each subset of $\mathbb{D}$, e.g., $\mathbf{C}_n$, is called a class in population game. IoTD makes the offloading decision (i.e., choosing optimal BS) based on network condition and task information. The aggregate offloading behaviors of IoTDs can be captured by class state and population state. We first define the class state as a distribution of the number of IoTDs choosing different BSs, denoted by vector $\boldsymbol{a}^n=[a_m^n]_{m\in\mathcal{M}^n}$. Note that a_m^n represents the number of IoTDs offloading tasks from $\mathbf{C}_n$ to $\mathbf{B}_m$. Then, we can represent the population state (i.e., the offloading decisions of all IoTDs) with the class states. The population state $\boldsymbol{a}=[\boldsymbol{a}^n]_{n\in\mathcal{N}}$ is a Cartesian product of all class states. Table I summarizes the basic notation used in the paper. We next consider the partition rule for $\mathbb{C}$.

A. IoT Device Classification

We classify IoTDs into different classes according to their locations and task information. The location of $\mathbf{D}_q$ is denoted by L_q, where L_q belongs to Cartesian plane $\mathbb{R}^2$. Assume

that the length of data flow from $\mathbf{D}_q$ follows an exponential distribution with average value B_q, the length of computation flow from $\mathbf{D}_q$ follows an exponential distribution with average value E_q and the task generation rate from $\mathbf{D}_q$ follows a Poisson Point Process with rate λ_q [12, 13]. We use $\mathbb{J}_q \triangleq (B_q, E_q, \lambda_q)$ to denote the task of $\mathbf{D}_q$. More specifically, B_q represents size of data including computational input data and execution codes. E_q denotes the required CPU cycles to execute task $\mathbb{J}_q$. Based on IoTD's location and task information, class $\mathbf{C}_n$ is defined as follows.

$$\mathbf{C}_n = \{q \in \mathcal{Q} \mid \frac{E_q}{B_q} = C^n, \lambda_q = \lambda^n, \text{ and } L_q = L^n\}. \quad (1)$$

IoTDs from class $\mathbf{C}_n$ should satisfy three conditions characterized by C^n, λ^n and L^n. C^n requires that IoTDs in $\mathbf{C}_n$ should have same computational density per unit size of data. λ^n ensures that IoTDs in $\mathbf{C}_n$ have the same task generation rate. L^n requires that IoTDs in $\mathbf{C}_n$ should be in the same location, since IoTDs in the same location face similar network environment (e.g., network traffic and link capacity). Note that IoTD can only offload mobile tasks to BSs in proximity. Let $\mathbb{B}^n = \{\mathbf{B}_m\}_{m \in \mathcal{M}^n}$ denote the available BSs that can execute tasks for $\mathbf{C}_n$. The number of IoTDs in $\mathbf{C}_n$ is denoted by Z^n. $\sum_{m \in \mathcal{M}^n} a_m^n = Z^n$ and $\sum_{n \in \mathcal{N}} Z^n = Q$ ensures that the class size and population size remains stable. As illustrated in Fig. 2, all IoTDs in a same class fall in a same line; the slope of the line denotes the computational density of the class.

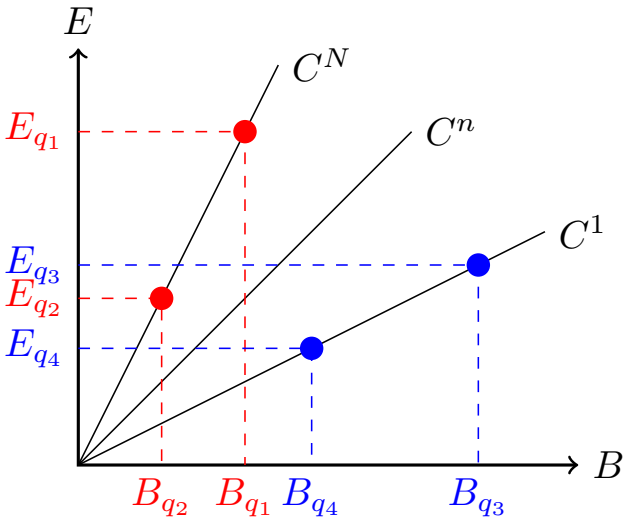

Fig. 2. Illustration of IoTDs classification model. We consider that four IoTDs $\mathbf{D}_{q_1}, \mathbf{D}_{q_2}, \mathbf{D}_{q_3}$ and $\mathbf{D}_{q_4}$ are located in the same place, i.e., $L_{q_1} = L_{q_2} = L_{q_3} = L_{q_4}$. Due to different computational density per size of data, e.g., $C^1 = \frac{E_{q_3}}{B_{q_3}} = \frac{E_{q_4}}{B_{q_4}}$ and $C^N = \frac{E_{q_1}}{B_{q_1}} = \frac{E_{q_2}}{B_{q_2}}$, IoTDs are classified into two classes. Note that each line can represent a class and the slope of the line denotes the computational density of the class. Thus, $q_1, q_2 \in \mathbf{C}_N$ and $q_3, q_4 \in \mathbf{C}_1$.

Population game requires that all IoTDs from the same class are homogeneous. Previous classification cannot preserve this property, since two IoTDs may have different data sizes even if they are in the same class. In order to solve this problem, we need to reconsider IoTDs' tasks and recalculate the class size. The basic idea is to divide the larger size data (and CPU cycles) into a number of minimum size data (and CPU cycles). We first select the IoTD with minimum data size (and corresponding minimum CPU cycles) in $\mathbf{C}_n$ as a benchmark, denoted as $\mathbf{D}_n^*$. Then, we consider that all the other IoTDs are composed of multiple $\mathbf{D}_n^*$s. For example, if the data size of $\mathbf{D}_n'$ is 1.5 times of that $\mathbf{D}_n^*$, then $\mathbf{D}_n'$ can be replaced by 1.5 $\mathbf{D}_n^*$s. Since all IoTDs in a class are replaced by $\mathbf{D}_n^*$, the homogeneous property is preserved. We can recalculate the class size and population size as follows:

$$\hat{Z}^n = \sum_{q \in \mathbf{C}_n} \frac{B_q}{B^n}, \qquad \hat{Q} = \sum_{n \in \mathcal{N}} \hat{Z}^n, \quad (2)$$

where $B^n = min_{q \in \mathbf{C}_n} B_q$ is the data size of $\mathbf{D}_n^*$. $\hat{Z}^n$ and $\hat{Q}$ denote the class size and population size after replacement, respectively. Note that $\hat{Z}^n$ may not be integer while Z^n is integer. Unless otherwise specified, we will use this new population model in the rest of the paper.

B. Task Execution Model

Tasks generated by IoTDs will be transferred to BSs and then executed in the corresponding edge cloud, as shown in Fig. 3. We assume that IoTDs in $\mathbf{C}_n$ generate tasks according to a Poisson Point Process with rate λ^n. We further assume that the data size and CPU cycles of $\mathbf{C}_n$ follow the exponential distributions with average values of B^n and E^n, respectively. Note that B^n and E^n correspond to the data size and CPU cycles of $\mathbf{D}_n^*$. We define the traffic generation density of $\mathbf{D}_n^*$ as $\theta^n = \lambda^n B^n$. Thus, the traffic generation density of $\mathbf{C}_n$ is $\hat{Z}^n \theta^n$, which is simply the multiplication of the number of IoTDs and the traffic generation density of $\mathbf{D}_n^*$. Similarly, we can define the computation generation density of $\mathbf{D}_n^*$ as $\eta^n = \lambda^n E^n$.

We first introduce the communication model between IoTDs in $\mathbf{C}_n$ and $\mathbf{B}_m$. We consider that IoTDs in the same class have the same data rate, while IoTDs in different classes can have different data rates. The data rate between $\mathbf{D}_n^*$ and $\mathbf{B}_m$ is defined as follows:

$$R_m^n(\boldsymbol{a}) = \frac{W_m}{a_m^n} \log_2 \left(1 + \frac{a_m^n P_l^n H_m^n}{\delta + \sum_{k \in \mathcal{N}_m^n} a_m^k P_l^k H_m^k}\right), \quad (3)$$

where

$$\mathcal{N}_m^n = \left\{k \in \mathcal{N} \setminus \{n\} : m \in \mathcal{M}^k\right\}, \quad a_m^n \neq 0.$$

W_m denotes the total bandwidth of $\mathbf{B}_m$. P_l^n and P_l^k represent the average transmission power of IoTDs in $\mathbf{C}_n$ and $\mathbf{C}_k$, respectively. H_m^n and H_m^k are the average channel gain between $\mathbf{B}_m$ and IoTDs in $\mathbf{C}_n$ and $\mathbf{C}_k$, respectively. We use δ to denote the noise power. The interference from other classes is $\sum_{k \in \mathcal{N}_m^n} a_m^k P_l^k H_m^k$, where $\mathcal{N}_m^n$ denotes the set of classes whose available BSs include $\mathbf{B}_m$. We don't consider

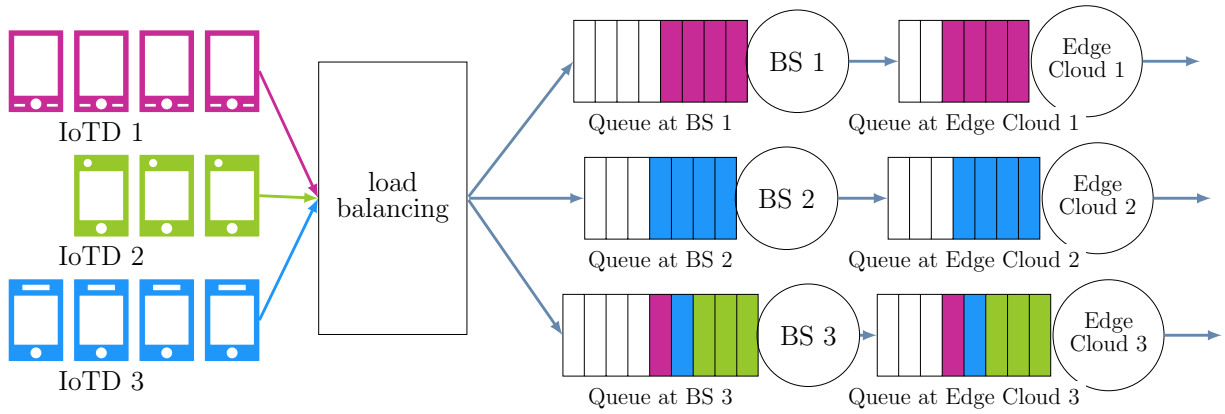

Fig. 3. Queueing model for MEC workload balancing. This figure illustrates that three classes of IoTDs offload tasks to three edge clouds. The processing delay consists of transmission delay in BS and computation delay in edge cloud. Load balancing mechanism can shorten the processing delay.

interference among IoTDs in a same class, since these IoTDs may come from a single IoTD before replacement. If no IoTD in $\mathbf{C}_n$ selects $\mathbf{B}_m$, i.e., $a_m^n = 0$, there is no need to calculate $R_m^n(\boldsymbol{a})$. Thus, we assume that $a_m^n \neq 0$ in Eq. (3).

The traffic load density of $\mathbf{B}_m$ serving an IoTD from $\mathbf{C}_n$ is defined as $\dot{T}_m^n(\boldsymbol{a}) = \frac{\theta^n}{R_m^n(\boldsymbol{a})}$, and denotes the time fraction of $\mathbf{B}_m$ serving $\mathbf{C}_n$. The utilization of $\mathbf{B}_m$ is the aggregation of traffic load density of $\mathbf{B}_m$ serving $\mathbf{C}_n$, which is defined as

$$\begin{aligned}\dot{\rho}_m(\boldsymbol{a}) &= \sum_{n\in\mathcal{N}} \dot{T}_m^n(\boldsymbol{a}) a_m^n = \sum_{n\in\mathcal{N}} \frac{\theta^n a_m^n}{R_m^n(\boldsymbol{a})} \\ &= \sum_{n\in\mathcal{N}} \frac{\theta^n (a_m^n)^2}{W_m \log_2\left(1 + \frac{a_m^n P_l^n H_m^n}{\delta + \sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k}\right)}. \end{aligned} \tag{4}$$

We then introduce the computation model when offloading tasks from $\mathbf{C}_n$ to $\mathbf{E}_m$. The computation load density of $\mathbf{E}_m$ (connected to $\mathbf{B}_m$) serving an IoTD from $\mathbf{C}_n$ is defined as $\ddot{T}_m^n = \frac{\eta^n}{F_m}$, where F_m is the computational capability (in CPU cycles/second) of $\mathbf{E}_m$. $\ddot{T}_m^n$ represents the time fraction of $\mathbf{E}_m$ serving $\mathbf{C}_n$. The utilization of $\mathbf{E}_m$ is the aggregation of computation load density of $\mathbf{E}_m$ serving $\mathbf{C}_n$, which is defined as

$$\ddot{\rho}_m = \sum_{n\in\mathcal{N}} \ddot{T}_m^n a_m^n = \sum_{n\in\mathcal{N}} \frac{\eta^n a_m^n}{F_m}. \tag{5}$$

C. Workload Balancing Model

The utilization levels of $\mathbf{B}_m$ and $\mathbf{E}_m$ are described by $\dot{\rho}_m(\boldsymbol{a})$ and $\ddot{\rho}_m$, respectively. In order to implement different load balancing for BSs and edge clouds, we take advantage of α-fair utility function [14] that we will maximize as follows:

$$\mathbf{T}(\alpha, \boldsymbol{\rho}) = \begin{cases} -\sum_{m\in\mathcal{M}} \frac{(1-\rho_m)^{1-\alpha} - 1}{\alpha - 1}, & \alpha \neq 1, \\ -\sum_{m\in\mathcal{M}} \ln\left(\frac{1}{1-\rho_m}\right), & \alpha = 1, \end{cases} \tag{6}$$

where $\boldsymbol{\rho} = \{\rho_m\}_{m\in\mathcal{M}}$ denotes the utilization status of BSs (when $\rho_m = \dot{\rho}_m(\boldsymbol{a})$) or edge clouds (when $\rho_m = \ddot{\rho}_m$). The load balancing factor α can have four different values resulting in four load balancing policies. For example, if $\alpha = 0$, then $\mathbf{T}(\alpha, \boldsymbol{\rho}) = \sum_{m\in\mathcal{M}} \rho_m$. The offloading decision is only based on IoTDs' perspective and this policy is called rate-optimal policy. If $\alpha = 2$, then $\mathbf{T}(\alpha, \boldsymbol{\rho}) = -\sum_{m\in\mathcal{M}} \frac{\rho_m}{1-\rho_m}$. Note that $\frac{\rho_m}{1-\rho_m}$ can represent the length of queue in $\mathbf{B}_m$ or $\mathbf{E}_m$. The negative sign is used to maximize α-fair utility function, since we aim to minimize the total length of queue $\sum_{m\in\mathcal{M}} \frac{\rho_m}{1-\rho_m}$. When $\alpha = 2$, $\mathbf{T}(\alpha, \boldsymbol{\rho})$ is called delay-optimal policy. Moreover, $\alpha = 1$ and $\alpha = \infty$ denote throughput-optimal policy and equalizing-load policy, respectively [14]. The authors in [15] show that $\alpha \geq 0$ can take more values except for the above cases. Thus, we can implement many kinds of load balancing in BSs and edge clouds by using different values of α.

III. Population Game Based Workload Balancing

In this section, we first propose a social welfare maximization problem that can implement efficient load balancing in BSs and edge clouds. Then, we define the payoff function of population game and introduce three basic evolutionary dynamics that can capture the evolution of population state.

A. Social Welfare Maximization

Our social welfare maximization aims to jointly implement load balancing in BSs and edge clouds and is defined as follows:

$$\max \quad \mathbf{T}(\dot{\alpha}, \ddot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a}), \ddot{\boldsymbol{\rho}}) = \mathbf{T}(\dot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a})) + \xi \mathbf{T}(\ddot{\alpha}, \ddot{\boldsymbol{\rho}}) \tag{7}$$

$$\text{s.t.} \quad \dot{\rho}_m(\boldsymbol{a}) < 1 \qquad \forall m \in \mathcal{M} \tag{8}$$

$$\ddot{\rho}_m < 1 \qquad \forall m \in \mathcal{M} \tag{9}$$

$$\sum_{m\in\mathcal{M}^n} a_m^n = \hat{Z}^n \qquad \forall n \in \mathcal{N}. \tag{10}$$

The $\dot{\alpha}$-fair utility function for BSs is denoted by $\mathbf{T}(\dot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a}))$. By replacing the input parameters of Eq. (6), we obtain that

$$\mathbf{T}(\dot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a})) = \begin{cases} -\sum\limits_{m\in\mathcal{M}} \dfrac{(1-\dot{\rho}_m(\boldsymbol{a}))^{1-\dot{\alpha}}-1}{\dot{\alpha}-1}, & \dot{\alpha}\neq 1, \\ -\sum\limits_{m\in\mathcal{M}} \ln\left(\dfrac{1}{1-\dot{\rho}_m(\boldsymbol{a})}\right), & \dot{\alpha}=1. \end{cases} \tag{11}$$

Similarly, we can obtain the $\ddot{\alpha}$-fair utility function for edge clouds as follows:

$$\mathbf{T}(\ddot{\alpha}, \ddot{\boldsymbol{\rho}}) = \begin{cases} -\sum\limits_{m\in\mathcal{M}} \dfrac{(1-\ddot{\rho}_m)^{1-\ddot{\alpha}}-1}{\ddot{\alpha}-1}, & \ddot{\alpha}\neq 1, \\ -\sum\limits_{m\in\mathcal{M}} \ln\left(\dfrac{1}{1-\ddot{\rho}_m}\right), & \ddot{\alpha}=1. \end{cases} \tag{12}$$

Since we jointly optimize the load balancing for BSs and edge clouds, $\xi > 0$ is a trade-off between these two objectives. Larger ξ implies higher priority in load balancing for edge clouds.

Constraints (8) and (9) indicate that the utilization of BSs and edge clouds can not exceed the maximum bandwidth and computational capability, respectively. Constraint (10) requires that the number of IoTDs in a class remains stable. Instead of solving this optimization problem directly, we propose a population game based method to solve the load balancing problem.

B. Population Game Formulation

In order to solve the optimization problem (7), we describe IoTDs' offloading decisions as a population state $\boldsymbol{a}$. The core of population game is the so-called payoff function. Payoff function defines IoTDs' payoffs based on a population state and is composed of a collection of marginal payoff functions, i.e., $\boldsymbol{F}(\boldsymbol{a}) = \{F_m^n(\boldsymbol{a}) : m \in \mathcal{M}^n, n \in \mathcal{N}\}$, $F_m^n(\boldsymbol{a})$ denotes the payoff of IoTD in $\mathbf{C}_n$ offloading task to $\mathbf{B}_m$ and is defined as follows:

$$F_m^n(\boldsymbol{a}) = -\left[\frac{\eta^n\xi}{F_m(1-\ddot{\rho}_m)^{\ddot{\alpha}}} + \frac{\theta^n\Big(2-g(\widehat{SINR}_m^n)\Big)}{R_m^n(\boldsymbol{a})(1-\dot{\rho}_m(\boldsymbol{a}))^{\dot{\alpha}}}\right], \tag{13}$$

where

$$g(\widehat{SINR}_m^n) = \frac{\widehat{SINR}_m^n}{(\widehat{SINR}_m^n+1)\ln(\widehat{SINR}_m^n+1)}, \tag{14}$$

$$\widehat{SINR}_m^n = \frac{a_m^n P_l^n H_m^n}{\delta + \sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k}. \tag{15}$$

$\widehat{SINR}_m^n$ denotes the Signal-to-Interference-plus-Noise Ratio (SINR) between $\mathbf{B}_m$ and $\mathbf{D}_m^n$. $g(\widehat{SINR}_m^n)$, called inference function, represents the effect of $\widehat{SINR}_m^n$ on load balancing among BSs. To better understand the definition of $F_m^n(\boldsymbol{a})$, we consider one simple case where $\dot{\alpha} = \ddot{\alpha} = 0$. In this case,

$$\hat{F}_m^n(\boldsymbol{a}) = -\left[\frac{\eta^n\xi}{F_m} + \frac{\theta^n\Big(2-g(\widehat{SINR}_m^n)\Big)}{R_m^n(\boldsymbol{a})}\right].$$

Recall that $\dot{T}_m^n(\boldsymbol{a}) = \frac{\theta^n}{R_m^n(\boldsymbol{a})}$ denotes the time fraction of $\mathbf{B}_m$ serving $\mathbf{C}_n$ and $\ddot{T}_m^n = \frac{\eta^n}{F_m}$ is the time fraction of $\mathbf{E}_m$ serving $\mathbf{C}_n$ (see Section II-B). We further obtain

$$\hat{F}_m^n(\boldsymbol{a}) = -\left[\xi\ddot{T}_m^n + \dot{T}_m^n(\boldsymbol{a})\Big(2-g(\widehat{SINR}_m^n)\Big)\right],$$

which has a similar structure of Eq. (7) except that the effect of SINR is obvious now. $\hat{F}_m^n(\boldsymbol{a})$ is the payoff of IoTDs from $\mathbf{C}_n$ choosing $\mathbf{B}_m$ and $\mathbf{E}_m$ (or the payoff of a_m^n). We observe that the time for transmission is affected by SINR and the loads of BSs and edge clouds are not considered in this case. When $\dot{\alpha} > 0$ and $\ddot{\alpha} > 0$, the load of BSs and edge clouds will affect the payoff of IoTDs, since $\ddot{\rho}_m$ and $\dot{\rho}_m(\boldsymbol{a})$ will be reserved in payoff function. Moreover, we propose theorem 1 to analyze the effect of $\Big(2-g(\widehat{SINR}_m^n)\Big)$ in payoff function.

Theorem 1. *Time fraction* $\hat{T}_m^n(\boldsymbol{a}) = \frac{\theta^n}{R_m^n(\boldsymbol{a})}\Big(2-g(\widehat{SINR}_m^n)\Big)$ *increases from* $\frac{\theta^n}{R_m^n(\boldsymbol{a})}$ *to* $\frac{2\theta^n}{R_m^n(\boldsymbol{a})}$*, when* $\widehat{SINR}_m^n$ *increases from* 0 *to* $+\infty$.

Theorem 1 implies that when $\widehat{SINR}_m^n = 0$, $\hat{T}_m^n(\boldsymbol{a}) = \dot{T}_m^n(\boldsymbol{a})$. As $\widehat{SINR}_m^n$ increases, time fraction $\hat{T}_m^n(\boldsymbol{a})$ increases. This is because higher $\widehat{SINR}_m^n$ implies higher data rate from IoTDs in $\mathbf{C}_n$; thus resulting in higher bandwidth utilization of $\mathbf{B}_m$. However, $\hat{T}_m^n(\boldsymbol{a})$ should be less than $\frac{2\theta^n}{R_m^n(\boldsymbol{a})}$, even if the transmission power is much larger than inference and noise power. The proof of Theorem 1 is given in Appendix A.

Definition 1. *A population game* $\boldsymbol{F} : \mathbb{R}_+^{N\times M} \to \mathbb{R}^{N\times M}$ *is a potential game if there exists a continuously differentiable function* $\mathbf{T} : \mathbb{R}_+^{N\times M} \to \mathbb{R}$*, called a potential function, satisfying* $\nabla\mathbf{T}(\boldsymbol{a}) = \boldsymbol{F}(\boldsymbol{a})$ *for all* $\boldsymbol{a} \in \mathbb{R}_+^{N\times M}$*, or* $\frac{\partial\mathbf{T}}{\partial a_m^n}(\boldsymbol{a}) = F_m^n(\boldsymbol{a})$ *for all* $m \in \mathcal{M}$ *and* $n \in \mathcal{N}$.

Definition 1 shows that the partial derivatives of the potential function are the payoff functions of the population game.

Theorem 2. *Our proposed population game* $\boldsymbol{F}(\boldsymbol{a}) = \{F_m^n(\boldsymbol{a}) : m \in \mathcal{M}^n, n \in \mathcal{N}\}$ *is a potential game. The potential function is* $\mathbf{T}(\dot{\alpha}, \ddot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a}), \ddot{\boldsymbol{\rho}})$.

Potential game always reaches an NE and has the finite improvement property. The proof of Theorem 2 is given in Appendix B.

C. Evolutionary Dynamics

NE is the solution concept of population game. By using the framework of evolutionary dynamics [16], we can analyze how population state evolves in time and converges to NE. The evolutionary dynamics is defined as follows:

$$\dot{a_m^n} = \sum_{k\in\mathcal{M}^n} a_k^n \rho_{km}^n(\boldsymbol{a}, \boldsymbol{F}(\boldsymbol{a})) - a_m^n \sum_{k\in\mathcal{M}^n} \rho_{mk}^n(\boldsymbol{a}, \boldsymbol{F}(\boldsymbol{a})), \tag{16}$$

where$\rho_{km}^n(\boldsymbol{a}, \boldsymbol{F}(\boldsymbol{a}))$, called revision protocol, represents the switching rate of IoTDs in $\mathbf{C}_n$ change offloading decision from $\mathbf{B}_k$ to $\mathbf{B}_m$ based on population state $\boldsymbol{a}$ and payoff function $\boldsymbol{F}(\boldsymbol{a})$. Larger value of $\rho_{km}^n(\boldsymbol{a}, \boldsymbol{F}(\boldsymbol{a}))$ implies higher probability that IoTDs in $\mathbf{C}_n$ changing offloading decision from $\mathbf{B}_k$ to $\mathbf{B}_m$. The first term and second term of Eq. (16) denote the inflow rate of IoTDs choosing $\mathbf{B}_m$ and the outflow rate of IoTDs choosing any BS except $\mathbf{B}_m$, respectively. Thus, the difference of inflow rate and outflow rate describes the evolution of a_m^n.

We consider three types of revision protocols, namely Smith, Logit and BNN [11, 16]. For simplicity, we use $\mathbf{D}_m^n$ to represent an IoTD in $\mathbf{C}_n$ choosing $\mathbf{B}_m$. Thus, a_m^n is the number of $\mathbf{D}_m^n$s. The function $[x]_+$ returns x if $x \geq 0$. Otherwise, it returns 0. Smith protocol, defined in Eq. (17), describes that the switching rate of $\mathbf{D}_k^n$ changing current offloading decision from $\mathbf{B}_k$ to $\mathbf{B}_m$ is the payoff difference between $\mathbf{D}_m^n$ and $\mathbf{D}_k^n$. For example, if $\mathbf{D}_k^n$ knows that $\mathbf{D}_m^n$ has higher payoff, i.e., $\mathbf{B}_m$ is a better choice than $\mathbf{B}_k$ for $\mathbf{C}_n$, then $\mathbf{D}_k^n$ will change his offloading decision to $\mathbf{B}_m$ with switching rate $\rho_{km}^n(\boldsymbol{a}, \boldsymbol{F}(\boldsymbol{a}))$. $\mathbf{D}_k^n$ will not change his offloading decision if $\mathbf{D}_m^n$ has lower payoff.

$$\rho_{km}^n(\boldsymbol{a}, \boldsymbol{F}(\boldsymbol{a})) = \left[F_m^n(\boldsymbol{a}) - F_k^n(\boldsymbol{a})\right]_+. \tag{17}$$

Logit protocol is defined in Eq. (18), where $\omega > 0$ is the noise level. ω represents the rationality of IoTDs. For $\omega = 0$, IoTDs are completely rational and choose the best offloading decision. As ω increases, IoTDs become less rational and may choose non-optimal decision.

$$\rho_{km}^n(\boldsymbol{a}, \boldsymbol{F}(\boldsymbol{a})) = \frac{\exp\left(\omega^{-1} F_m^n(\boldsymbol{a})\right)}{\sum_{k\in\mathcal{M}^n} \exp\left(\omega^{-1} F_k^n(\boldsymbol{a})\right)}. \tag{18}$$

BNN protocol, defined in Eq. (19), describes that $\mathbf{D}_k^n$ compares $\mathbf{D}_m^n$'s payoff with the average payoff of $\mathbf{C}_n$. If $\mathbf{D}_m^n$'s payoff exceeds the average payoff, then $\mathbf{D}_k^n$ will change his offloading decision to $\mathbf{B}_m$ with switching rate $\rho_{km}^n(\boldsymbol{a}, \boldsymbol{F}(\boldsymbol{a}))$.

$$\rho_{km}^n(\boldsymbol{a}, \boldsymbol{F}(\boldsymbol{a})) = \left[F_m^n(\boldsymbol{a}) - \frac{1}{\hat{Z}^n} \sum_{l\in\mathcal{M}^n} a_l^n F_l^n(\boldsymbol{a})\right]_+. \tag{19}$$

Note that these protocols describe how IoTDs change their offloading decisions until an NE is reached. Smith uses less decision information compared to Logit and BNN. Smith needs only the payoff of one BS, while Logit and BNN need the payoffs of all BSs in $\mathbb{B}^n$. In general, Smith has lower convergence speed than Logit and BNN. By substituting these revision protocols into Eq. (16), we can get Smith dynamics, Logit dynamics and BNN dynamics in Eqs. (20), (21) and (22), respectively.

$$\dot{a_m^n} = \sum_{k\in\mathcal{M}^n} a_k^n \left[F_m^n(\boldsymbol{a}) - F_k^n(\boldsymbol{a})\right]_+ - a_m^n \sum_{k\in\mathcal{M}^n} \left[F_k^n(\boldsymbol{a}) - F_m^n(\boldsymbol{a})\right]_+. \tag{20}$$

$$\dot{a_m^n} = \frac{a_m^n \exp\left(\omega^{-1} F_m^n(\boldsymbol{a})\right)}{\sum_{k\in\mathcal{M}^n} a_k^n \exp\left(\omega^{-1} F_k^n(\boldsymbol{a})\right)} - a_m^n. \tag{21}$$

$$\dot{a_m^n} = \left[\hat{F}_m^n(\boldsymbol{a})\right]_+ - \frac{a_m^n}{\hat{Z}^n} \sum_{k\in\mathcal{M}^n} \left[\hat{F}_k^n(\boldsymbol{a})\right]_+, \tag{22}$$

$$\text{where } \hat{F}_m^n(\boldsymbol{a}) = F_m^n(\boldsymbol{a}) - \frac{1}{\hat{Z}^n} \sum_{l\in\mathcal{M}^n} a_l^n F_l^n(\boldsymbol{a}).$$

These evolutionary dynamics can generate an NE in iteration methods. We will propose a load balancing algorithm based on evolutionary dynamics in next section.

IV. Workload Balancing Algorithms

In this section, we proposes two workload balancing algorithms, namely, CWB (Centralized Workload Balancing) algorithm and DWB (Decentralized Workload Balancing) algorithm. CWB is based on the evolutionary dynamics and DWB is based on Theorem 2.

A. Centralized Workload Balancing

Our CWB algorithm consists of two phases. The first phase (Steps $1-10$) is to calculate an NE based on evolutionary dynamics. Smith, Logit and BNN dynamics can converge to an NE with different convergence speeds (as shown in Section V-A). However, the resulting NE only shows the number of IoTDs in $\mathbf{C}_n$, that will choose $\mathbf{B}_m$, i.e., a_m^n, without specifying which IoTD in $\mathbf{C}_n$ will choose $\mathbf{B}_m$. Thus, we use the second phase (Steps $11-23$) to implement IoTDs' offloading decisions. The basic idea is to randomly select IoTDs from $\mathbf{C}_n$ for $\mathbf{B}_m$; the number of IoTDs should be no more than a_m^n. The randomness of selection can implement fair offloading decisions for IoTDs. Note that we need to replace $\mathbf{D}_n^*$ with original IoTD when calculating offloading decisions (Steps $18-20$).

B. Decentralized Workload Balancing

Our DWB algorithm is a distributed algorithm consisting of two parts of algorithms running in IoTDs and BSs separately. Each IoTD can change his current offloading decision whenever "stochastic update clock" rings. IoTD randomly chooses several candidate BSs and choose the BS with highest switching rate. IoTD only changes his offloading decision when the highest switching rate is positive, which implies higher payoff. With the property of potential game, any better update of offloading decision is guaranteed to reach an NE. Note that IoTD's algorithm (Part 1 in Algorithm 2) does not exactly follow the switching rates of Smith, Logit and BNN protocols. However, it captures the main features of these revision protocols: 1) randomness is ensured by Step 2, K can

Algorithm 1 CWB (Centralized Workload Balancing)

Phase 1: *Calculate an NE.*
Initialize $\boldsymbol{a}$ with arbitrary value satisfying Constraint (10)
$\bar{\boldsymbol{a}} \leftarrow \mathbf{0}$
while $\bar{\boldsymbol{a}} \neq \boldsymbol{a}$ **do**
  $\bar{\boldsymbol{a}} \leftarrow \boldsymbol{a}$
  for all $n \in \mathcal{N}$ **do**
    for all $m \in \mathcal{M}^n$ **do**
      $a_m^n \leftarrow$ Update $\bar{a}_m^n$ with Eqs. (20), (21) or (22)
    end for
  end for
end while
Phase 2: *Calculate offloading decisions.*
Initialize decision vector $\mathbf{A} \leftarrow \mathbf{0}$
for all $n \in \mathcal{N}$ **do**
  $\mathbf{C} \leftarrow \mathbf{C}^n$
  for all $m \in \mathcal{M}^n$ **do**
    $a \leftarrow a_m^n$
    while $\mathbf{C} \neq \varnothing \wedge a > 0$ **do**
      $i \leftarrow Next(\mathbf{C})$
      if $\frac{B_i}{B^n} \leq a$ **then**
        $a \leftarrow a - \frac{B_i}{B^n}$, $\mathbf{C} \leftarrow \mathbf{C} \setminus \{i\}$, $\mathbf{A}(i) \leftarrow m$
      end if
    end while
  end for
end for

be any value between 1 and $|\mathbb{B}^n|$, we choose $K = \left\lfloor \frac{|\mathbb{B}^n|}{2} \right\rfloor$; 2) higher switching rate implying higher probability of changing offloading decision is guaranteed by Steps $3-9$. BS's algorithm (Part 2 in Algorithm 2) is to first collect current IoTDs' offloading decisions and then broadcast the utilization level of BSs and edge clouds to IoTDs. IoTDs will use this information to improve their offloading decisions.

V. Performance Evaluation

In this section, we evaluate the performance of our proposed workload balancing algorithms by numerical studies. We compare our work with other two contributions: BRUTE [12] and TWB (Towards Workload Balancing) [13]. BRUTE uses $\alpha-$fair function to implement energy-efficient traffic allocation among BSs. BRUTE considers the energy consumption in BSs without considering the load in edge clouds. TWB considers the traffic load in BSs and computation load in edge clouds. However, none of them considers the effect of SINR in load balancing.

Without loss of generality, We randomly select the parameter's value from the normal distribution for different cases. The average value of these parameters are illustrated as in [17–20]. The data size B^n is set to $800KB$ and the number of required CPU cycles D^n is set to $1000Megacycles$. We set the allocated computational capability F_m^n to $100GHz$ and the bandwidth of BS to $5MHz$. The channel gain between $\mathbf{D}_n$ and $\mathbf{B}_m$ is $H_m^n = (d_n^m)^{-\theta}$, where θ is the pass loss factor and d_n^m is the distance between them. θ is set to 4 and H_m^n

Algorithm 2 DWB (Decentralized Workload Balancing)

Part 1: *For all* $\mathbf{D}_m^n$
for all "stochastic update clock" rings **do**
  $\mathcal{K} \leftarrow$ randomly choose K BSs from $\mathbb{B}^n$
  for all $k \in \mathcal{K}$ **do**
    Calculate $\rho_{mk}^n(\hat{\boldsymbol{a}}, \hat{\boldsymbol{F}}(\boldsymbol{a}))$ with Eqs. (17), (18) or (19)
  end for
  if $\max_{k \in \mathcal{K}} \rho_{km}^n(\hat{\boldsymbol{a}}, \hat{\boldsymbol{F}}(\boldsymbol{a})) > 0$ **then**
    $k^* = \arg\max_{k \in \mathcal{K}} \rho_{km}^n(\hat{\boldsymbol{a}}, \hat{\boldsymbol{F}}(\boldsymbol{a}))$
    Update the offloading decision of $\mathbf{D}_m^n$ with k^*
  end if
end for
Part 2: *For all* $\mathbf{B}_m$
for all "stochastic update clock" rings **do**
  Collect current population state $\hat{\boldsymbol{a}}$
  Calculate $\hat{\hat{\rho}}_m$ and $\hat{\hat{\rho}}_m$ by Eqs. (4) and (5), respectively.
  Broadcast $\hat{\hat{\rho}}_m$ and $\hat{\hat{\rho}}_m$ to IoTDs within coverage of $\mathbf{B}_m$
end for

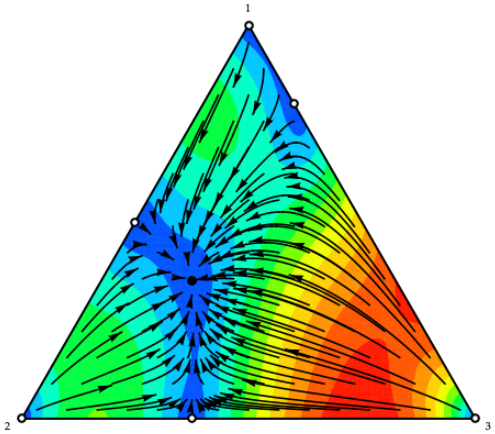

Fig. 4. Evolutionary dynamics of Smith protocol. The black dot denotes the NE. The arrows describe the motions of different population states.

is randomly selected from $[5m - 100m]$ [21]. The wireless transmission power P_l^n is set to $100mWatts$.

A. Illustration of Evolutionary Dynamics: One Class Case

We first compare the evolutionary dynamics of Smith, Logit and BNN. We consider the scenario where a class of 400 IoTDs compete for the communication resources of 3 BSs and computation resources in 3 edge clouds. We observe that Smith converges more slowly than Logit and BNN, as shown in Figs. 4, 5 and 6. We also observe that the arrows of Smith approach NE in a less angular, more gradual fashion. This is because Smith changes its offloading decision based on the payoff of one BS, while BNN and Logit change the offloading

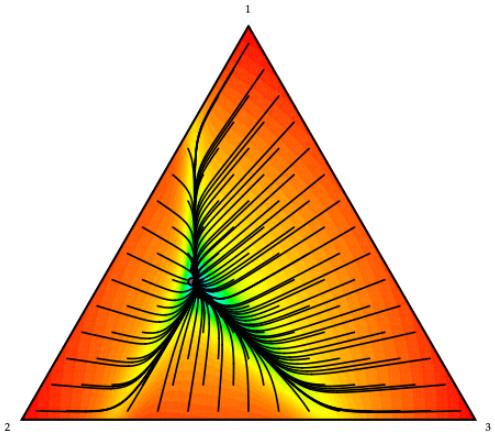

Fig. 5. Evolutionary dynamics of Logit protocol. The black dot denotes the NE. The arrows describe the motions of different population states.

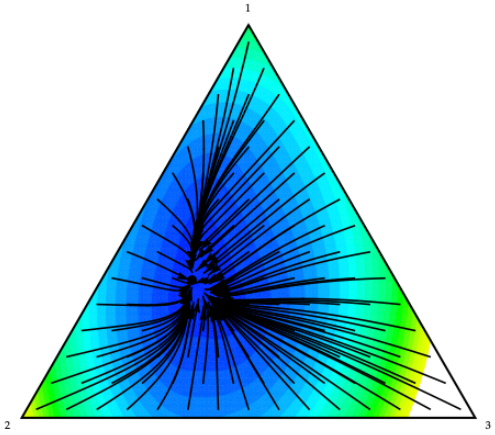

Fig. 6. Evolutionary dynamics of BNN dynamics. The black dot denotes the NE. The arrows describe the motions of different population states.

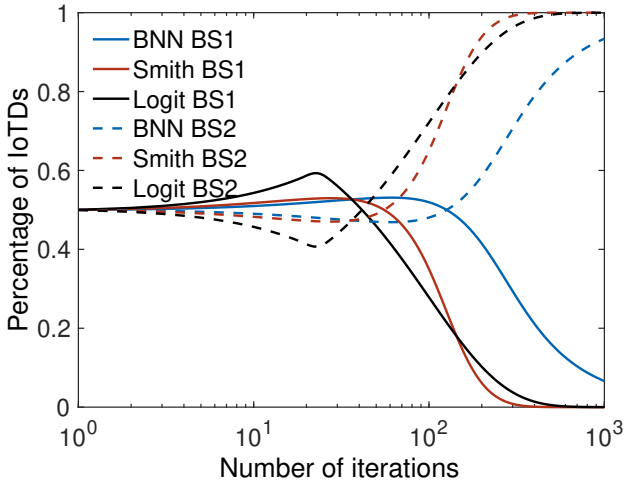

Fig. 7. Percentage of IoTDs in $\mathbf{C}_1$ choosing $\mathbf{B}_1$ and $\mathbf{B}_2$ with respect to the number of iterations. The initial class state $\boldsymbol{a}^1 = (0.5, 0.5)$. The solid line represents the percentage of IoTDs choosing $\mathbf{B}_1$, while the dashed line represents the percentage of IoTDs choosing $\mathbf{B}_2$.

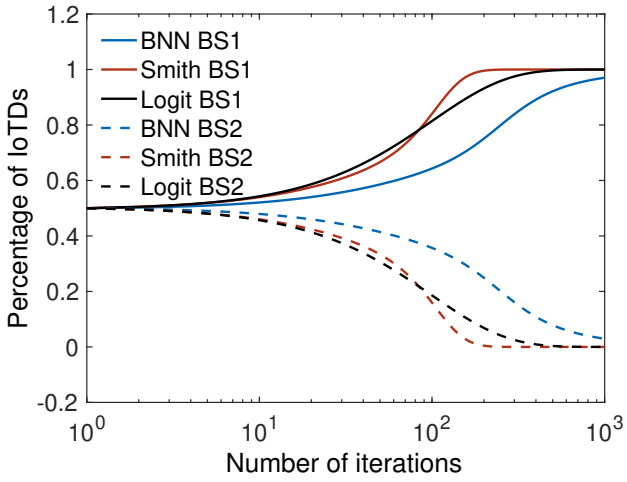

Fig. 8. Percentage of IoTDs in $\mathbf{C}_2$ choosing $\mathbf{B}_1$ and $\mathbf{B}_2$ with respect to the number of iterations. The initial class state $\boldsymbol{a}^2 = (0.5, 0.5)$.

decision based on all the payoff of BSs. Generally, using more payoff information to make offloading decisions can achieve better performance. Thus, BNN and Logit converge faster than Smith. Moreover, these three evolutionary dynamics can achieve the same NE.

Then, we investigate the resulting NE, i.e., the percentage of IoTDs choosing three BSs, denoted by black point in the figures. Note that the distance between black point and the vertex of triangle denotes the percentage of IoTDs choosing the corresponding BS; smaller distance represents higher percentage. NE is $(0.35, 0.45, 0.20)$ in our settings. We observe that NE is located close to BS 2 (i.e., the black point is close to vertex 2). This is because BS 2 and edge cloud 2 have the highest communication and computation capabilities in our settings. Consequently, more IoTDs prefer to offload tasks to $\mathbf{B}_2$.

B. Illustration of Evolutionary Dynamics: Two Classes Case

We consider the scenario where two classes of 400 IoTDs compete for the communication resources of 2 BSs and computation resources in 2 edge clouds. Figs. 7 and 8 show the evolutions of $\boldsymbol{a}^1$ and $\boldsymbol{a}^2$, respectively. The initial class states of $\boldsymbol{a}^1$ and $\boldsymbol{a}^2$ are both $(0.5, 0.5)$. The NE for $\mathbf{C}_1$ is $(0, 1)$ and the NE for $\mathbf{C}_2$ is $(1, 0)$; this shows that IoTDs in $\mathbf{C}_1$ will offload tasks to $\mathbf{B}_2$ and IoTDs in $\mathbf{C}_2$ will offload tasks to $\mathbf{B}_1$. We observe that three dynamics converge to NE with diverse convergence speeds. Smith has higher convergence speed than BNN and Logit. This is because Smith dynamic requires user to choose some BS with higher payoff. Since there are two BSs in this scenario, IoTDs using Smith dynamic

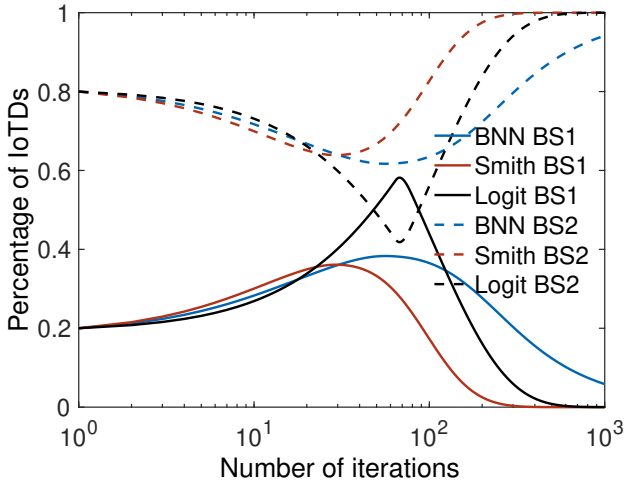

Fig. 9. Percentage of IoTDs in $\mathbf{C}_1$ choosing $\mathbf{B}_1$ and $\mathbf{B}_2$ with respect to the number of iterations. The initial class state $\boldsymbol{a}^1 = (0.2, 0.8)$.

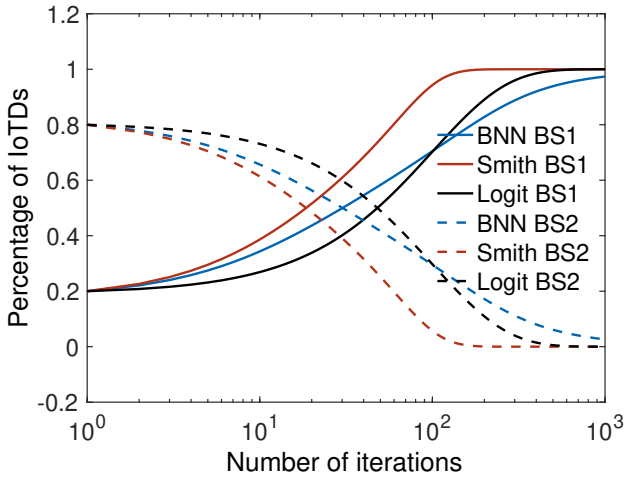

Fig. 10. Percentage of IoTDs in $\mathbf{C}_2$ choosing $\mathbf{B}_1$ and $\mathbf{B}_2$ with respect to the number of iterations. The initial class state $\boldsymbol{a}^2 = (0.2, 0.8)$.

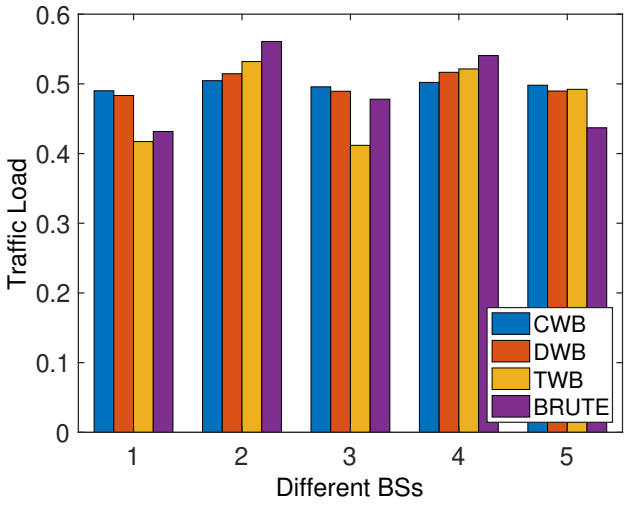

Fig. 11. Traffic load versus different BSs.

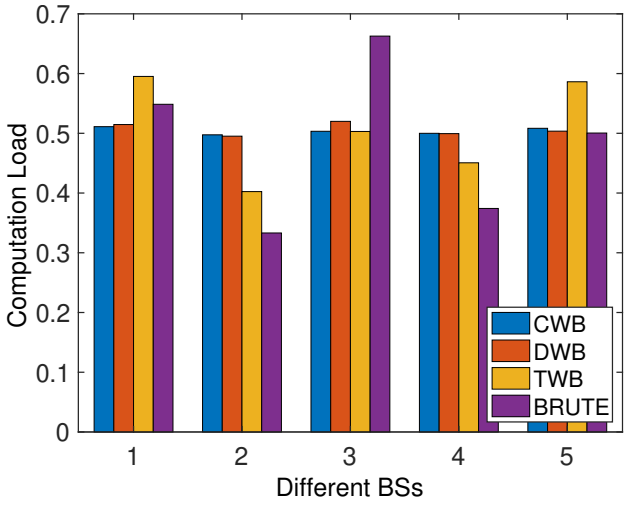

Fig. 12. Computation load versus different edge clouds.

has higher probability to choose optimal decisions. In BNN dynamic, IoTDs choose BS randomly and compare its payoff with average payoff in the same class; IoTDs have more chance to choose suboptimal decision compared to Smith. The same case happens to Logit dynamic where IoTDs change decisions according to the payoff ratios defined in Eq. (18). We state that the advantages of BNN and Logit increase with the number of BSs. The intuition is that IoTDs using Smith dynamic has lower probability to choose optimal decisions with the increase of the number of BSs, since Smith only ensures that the next payoff is better than current payoff, while BNN ensures that next payoff is better than average payoff and Logit uses the payoffs of all IoTDs in a class to make decisions.

Figs. 9 and 10 illustrate the evolutions of $\boldsymbol{a}^1$ and $\boldsymbol{a}^2$, respectively. We set the initial class states of $\boldsymbol{a}^1$ and $\boldsymbol{a}^2$ to $(0.2, 0.8)$, which is different from that in Figs. 7 and 8. We observe that three dynamics converge to NE. This demonstrates that diverse initial class states can reach the same NE.

C. Performance Comparison of Different Algorithms

To investigate the performance of workload balancing of different algorithms, we consider a scenario where 1000 IoTDs compete for communication and computation resources of 5 BS-edge-cloud pairs. Fig. 11 shows that CWB and DWB achieve better performance than TWB and BRUTE in traffic load balancing. We observe that the difference of traffic load among 5 BSs achieved by CWB and DWB are smaller than those achieved by TWB and BRUTE. This is because CWB and DWB use inference affected queueing model where SINR dynamically changes with population state, while TWB and BRUTE use static SINR model where SINR is estimated as location-dependent static value. We further observe that BRUTE achieves higher performance for traffic load balancing than TWB, since BRUTE does not consider the computation load balancing in edge clouds. Thus, BRUTE does not need

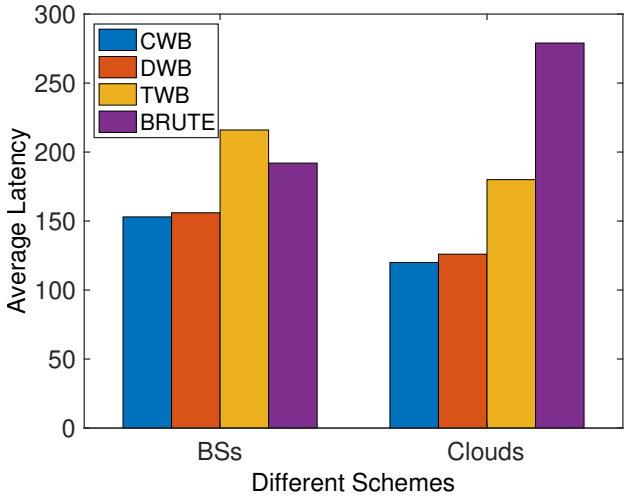

Fig. 13. Average latency versus different schemes.

to sacrifice the load balancing among BSs to implement load balancing among edge clouds. Fig. 12 shows the computation load among edge clouds. We observe that CWB and DWB achieve better performance than TWB and BRUTE in computation load balancing. This is because TWB and BRUTE do not consider the affect of dynamic inference among IoTDs. Furthermore, BRUTE achieves worst performance without considering the computation load balancing in edge clouds.

Fig. 13 shows the average latency in BSs and edge clouds with 4 schemes. We observe that CWB achieves lowest latency and DWB achieves quasi-optimal latency due to stochastic factor. Since BRUTE only focuses on communication latency among BSs, it has lower communication latency than TWB. In contrast, TWB achieves much lower computation latency than BRUTE, since TWB considers both communication latency and computation latency. By sacrificing slight communication latency, TWB achieves better performance than BRUTE. However, CWB and DWB outperform BRUTE and TWB, since BRUTE and TWB do not consider the impact of SINR.

VI. Conclusion

In this paper, we analyze the workload balancing problem for MEC in the context of IoT. IoTDs can offload computation intensive tasks to nearby BSs. We propose a population game based approach to investigate this problem and show that the game always has an NE. We consider the impact of SINR in workload balancing problem and propose an inference function to analyze the impact of SINR. We use three kinds of revision protocols, namely, BNN, Smith and Logit to achieve NE. We design two workload balancing algorithms to iteratively calculate the optimal solution. We illustrate the evolutionary dynamics with two different scenarios. Numerical results show that our schemes outperform two existing schemes.

References

[1] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for iot systems: A computation offloading game," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3246–3257, Aug 2018.

[2] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multi-access mobile edge computing in ultra-dense networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 14–19, 2018.

[3] H. Guo, J. Liu, and H. Qin, "Collaborative mobile edge computation offloading for iot over fiber-wireless networks," *IEEE Network*, vol. 32, no. 1, pp. 66–71, 2018.

[4] D. Liu, A. Hafid, and L. Khoukhi, "Population game based energy and time aware task offloading for large amounts of competing users," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec 2018, pp. 1–6.

[5] D. Liu, L. Khoukhi, and A. Hafid, "Prediction-based mobile data offloading in mobile cloud computing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 7, pp. 4660–4673, July 2018.

[6] ——, "Decentralized data offloading for mobile cloud computing based on game theory," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, May 2017, pp. 20–24.

[7] J. Zhang, W. Xia, Y. Zhang, Q. Zou, B. Huang, F. Yan, and L. Shen, "Joint offloading and resource allocation optimization for mobile edge computing," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[8] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[9] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Mobile edge computing and networking for green and low-latency internet of things," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 39–45, 2018.

[10] J. Barreiro-Gomez, G. Obando, and N. Quijano, "Distributed population dynamics: Optimization and control applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 2, pp. 304–314, 2017.

[11] N. Quijano, C. Ocampo-Martinez, J. Barreiro-Gomez, G. Obando, A. Pantoja, and E. Mojica-Nava, "The role of population games and evolutionary dynamics in distributed control systems: The advantages of evolutionary game theory," *IEEE Control Systems Magazine*, vol. 37, no. 1, pp. 70–97, 2017.

[12] S. Moon, H. Kim, and Y. Yi, "Brute: Energy-efficient user association in cellular networks from population game perspective," *IEEE Transactions on Wireless Communications*, vol. 15, no. 1, pp. 663–675, 2016.

[13] Q. Fan and N. Ansari, "Towards workload balancing in fog computing empowered iot," *IEEE Transactions on Network Science and Engineering*, 2018.

[14] H. Kim, G. De Veciana, X. Yang, and M. Venkatachalam, "Distributed α-optimal user association and cell load balancing in wireless networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 1, pp. 177–190, 2012.

[15] C. Guo, Y. Zhang, M. Sheng, X. Wang, and Y. Li, "α-fair power allocation in spectrum-sharing networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 5, pp. 3771–3777, 2016.

[16] W. H. Sandholm, "Population games and deterministic evolutionary dynamics," in *Handbook of game theory with economic applications*. Elsevier, 2015, vol. 4, pp. 703–778.

[17] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, "Energy aware offloading for competing users on a shared communication channel," *IEEE Transactions on Mobile Computing*, vol. 16, no. 1, pp. 87–96, 2017.

[18] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[19] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation Offloading for Service Workflow in Mobile Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3317–3329, 2015.

[20] D. Liu, L. Khoukhi, and A. Hafid, "Data offloading in mobile cloud computing: A markov decision process approach," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[21] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2015.

APPENDIX A
PROOF OF THEOREM 1

Proof. In order to proof Theorem 1, we need investigate the properties of inference function $g(\widehat{SINR}_m^n)$.

A. Inference Function is a Monotonic Function

We first calculate the derivative of $g(\widehat{SINR}_m^n)$ with respect to $\widehat{SINR}_m^n$ as follows:

$$g'(\widehat{SINR}_m^n) = \frac{(\widehat{SINR}_m^n+1)\ln(\widehat{SINR}_m^n+1)}{(\widehat{SINR}_m^n+1)^2\ln^2(\widehat{SINR}_m^n+1)} - \frac{\widehat{SINR}_m^n(\ln(\widehat{SINR}_m^n+1)+1)}{(\widehat{SINR}_m^n+1)^2\ln^2(\widehat{SINR}_m^n+1)} \tag{23}$$

$$= \frac{\ln(\widehat{SINR}_m^n+1)-\widehat{SINR}_m^n}{(\widehat{SINR}_m^n+1)^2\ln^2(\widehat{SINR}_m^n+1)}. \tag{24}$$

Note that the domain of $g(\widehat{SINR}_m^n)$ is $\{\widehat{SINR}_m^n \in \mathbb{R}|\widehat{SINR}_m^n > 0\}$, since $g(\widehat{SINR}_m^n)$ has no definition when $\widehat{SINR}_m^n = 0$. We will discuss the case where $\widehat{SINR}_m^n = 0$ later. Obviously, $\ln(\widehat{SINR}_m^n+1)-\widehat{SINR}_m^n < 0$ for all $\widehat{SINR}_m^n > 0$. According to Eq. (24), we know that $g'(\widehat{SINR}_m^n) < 0$. Thus, $g(\widehat{SINR}_m^n)$ is a strictly decreasing function.

B. The Limit of Inference Function

We next consider the value of $\lim_{\widehat{SINR}_m^n\to 0} g(\widehat{SINR}_m^n)$.

$$\lim_{\widehat{SINR}_m^n\to 0} g(\widehat{SINR}_m^n) \tag{25}$$

$$= \lim_{\widehat{SINR}_m^n\to 0} \frac{\widehat{SINR}_m^n}{(\widehat{SINR}_m^n+1)\ln(\widehat{SINR}_m^n+1)} \tag{26}$$

$$= \lim_{\widehat{SINR}_m^n\to 0} \frac{\widehat{SINR}_m^n}{\ln(\widehat{SINR}_m^n+1)} \lim_{\widehat{SINR}_m^n\to 0} \frac{1}{\widehat{SINR}_m^n+1} \tag{27}$$

$$= \lim_{\widehat{SINR}_m^n\to 0} \frac{\widehat{SINR}_m^n}{\ln(\widehat{SINR}_m^n+1)} \text{ (Applying l'Hopital's rule)} \tag{28}$$

$$= \lim_{\widehat{SINR}_m^n\to 0} \frac{\frac{d}{d\widehat{SINR}_m^n}(\widehat{SINR}_m^n)}{\frac{d}{d\widehat{SINR}_m^n}(\ln(\widehat{SINR}_m^n+1))} \tag{29}$$

$$= \lim_{\widehat{SINR}_m^n\to 0} (\widehat{SINR}_m^n+1) \tag{30}$$

$$= 1. \tag{31}$$

Without causing ambiguity, we say that $g(\widehat{SINR}_m^n) = 1$, when $\widehat{SINR}_m^n = 0$.

C. The Range of Inference Function

Since $g(\widehat{SINR}_m^n)$ is strictly decreasing and $g(0) = 1$, we know that

$$\max_{\widehat{SINR}_m^n\geq 0} g(\widehat{SINR}_m^n) = g(0) = 1. \tag{32}$$

Moreover, it is easy to verify that $g(\widehat{SINR}_m^n) > 0$ for all $\widehat{SINR}_m^n \geq 0$ and $\lim_{\widehat{SINR}_m^n\to+\infty} g(\widehat{SINR}_m^n) = 0$. Thus, the range of $g(\widehat{SINR}_m^n)$ is $[0, 1]$.

Based on the above discussions, we know that $g(\widehat{SINR}_m^n)$ decreases from 1 to 0 when $\widehat{SINR}_m^n$ increases from 0 to $+\infty$. Thus, $\left(2 - g(\widehat{SINR}_m^n)\right)$ increases from 1 to 2 when $\widehat{SINR}_m^n$ increases from 0 to $+\infty$. Finally, time fraction $\hat{T}_m^n(\boldsymbol{a}) = \frac{\theta^n}{R_m^n(\boldsymbol{a})}\left(2 - g(\widehat{SINR}_m^n)\right)$ increases from $\frac{\theta^n}{R_m^n(\boldsymbol{a})}$ to $\frac{2\theta^n}{R_m^n(\boldsymbol{a})}$, when $\widehat{SINR}_m^n$ increases from 0 to $+\infty$. □

APPENDIX B
PROOF OF THEOREM 2

Proof. In order to prove that $\mathbf{T}(\dot{\alpha}, \ddot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a}), \ddot{\boldsymbol{\rho}})$ is the potential function of population game $\boldsymbol{F}(\boldsymbol{a})$, we need to derive that

$$F_m^n(\boldsymbol{a}) = \frac{\partial \mathbf{T}(\dot{\alpha}, \ddot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a}), \ddot{\boldsymbol{\rho}})}{\partial a_m^n}. \tag{33}$$

A. Partial Derivatives of $\mathbf{T}(\ddot{\alpha}, \ddot{\boldsymbol{\rho}})$

We first calculate the partial derivatives of $\mathbf{T}(\ddot{\alpha}, \ddot{\boldsymbol{\rho}})$ in two cases based on the value of $\ddot{\alpha}$.

Case 1: $\ddot{\alpha} \neq 1$,

$$\frac{\partial \mathbf{T}(\ddot{\alpha}, \ddot{\boldsymbol{\rho}})}{\partial a_m^n} = \frac{\partial}{\partial a_m^n}\Big[-\sum_{m\in\mathcal{M}} \frac{(1-\ddot{\rho}_m)^{1-\ddot{\alpha}}-1}{\ddot{\alpha}-1}\Big] \tag{34}$$

$$= \frac{1-\ddot{\alpha}}{(\ddot{\alpha}-1)(1-\ddot{\rho}_m)^{\ddot{\alpha}}} \cdot \frac{\partial \ddot{\rho}_m}{\partial a_m^n} \tag{35}$$

$$= \frac{-\eta^n}{F_m(1-\ddot{\rho}_m)^{\ddot{\alpha}}}. \tag{36}$$

Based on Eq. (12), we have Eq. (34). According to Eq. (5), we can obtain that $\frac{\partial \ddot{\rho}_m}{\partial a_m^n} = \frac{\eta^n}{F_m}$. Thus, we can get Eq. (36) from Eq. (35). Similarly, we can calculate the partial derivatives of $\mathbf{T}(\ddot{\alpha}, \ddot{\boldsymbol{\rho}})$ when $\ddot{\alpha} = 1$.

Case 2: $\ddot{\alpha} = 1$,

$$\frac{\partial \mathbf{T}(\ddot{\alpha}, \ddot{\boldsymbol{\rho}})}{\partial a_m^n} = \frac{\partial}{\partial a_m^n}\Big[-\sum_{m\in\mathcal{M}} \ln\left(\frac{1}{1-\ddot{\rho}_m}\right)\Big] \tag{37}$$

$$= \frac{1-\ddot{\rho}_m}{(1-\ddot{\rho}_m)^2} \cdot \frac{-\partial \ddot{\rho}_m}{\partial a_m^n} \tag{38}$$

$$= \frac{-\eta^n}{F_m(1-\ddot{\rho}_m)}. \tag{39}$$

Note that Eq. (39) is equal to Eq. (36) when $\ddot{\alpha} = 1$. Thus, we can say that

$$\frac{\partial \mathbf{T}(\ddot{\alpha}, \ddot{\boldsymbol{\rho}})}{\partial a_m^n} = \frac{-\eta^n}{F_m(1-\ddot{\rho}_m)^{\ddot{\alpha}}}, \qquad \forall \ddot{\alpha} \geq 0. \tag{40}$$

B. Partial Derivatives of $\mathbf{T}(\dot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a}))$

We next calculate the partial derivatives of $\mathbf{T}(\dot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a}))$ based on different values of $\dot{\alpha}$. Case 1: $\dot{\alpha} \neq 1$,

$$\frac{\partial \dot{\rho}_m(\boldsymbol{a})}{\partial a_m^n} = \frac{\partial}{\partial a_m^n}\left[\sum_{n\in\mathcal{N}} \frac{\theta^n (a_m^n)^2}{W_m \log_2\left(1+\frac{a_m^n P_l^n H_m^n}{\delta+\sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k}\right)}\right] = \frac{\theta^n \ln(2)}{W_m}\cdot\frac{\partial}{\partial a_m^n}\left[\frac{(a_m^n)^2}{\ln\left(1+\frac{a_m^n P_l^n H_m^n}{\delta+\sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k}\right)}\right] \tag{41}$$

$$= \frac{\theta^n \ln(2)\left(2a_m^n \ln\left(\frac{a_m^n P_l^n H_m^n}{\sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k+\delta}+1\right) - \frac{(a_m^n)^2 P_l^n H_m^n}{(\sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k+\delta)\left(\frac{a_m^n P_l^n H_m^n}{\sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k+\delta}+1\right)}\right)}{W_m \ln^2\left(\frac{a_m^n P_l^n H_m^n}{\sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k+\delta}+1\right)} \tag{42}$$

$$= \frac{\theta^n\left(2a_m^n \log_2\left(\frac{a_m^n P_l^n H_m^n}{\sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k+\delta}+1\right) - \frac{(a_m^n)^2 P_l^n H_m^n}{\ln(2)(\sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k+\delta)\left(\frac{a_m^n P_l^n H_m^n}{\sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k+\delta}+1\right)}\right)}{W_m \log_2^2\left(\frac{a_m^n P_l^n H_m^n}{\sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k+\delta}+1\right)}. \tag{43}$$

$$\frac{\partial \mathbf{T}(\dot{\alpha},\dot{\boldsymbol{\rho}}(\boldsymbol{a}))}{\partial a_m^n} = \frac{\partial}{\partial a_m^n}\Big[-\sum_{m\in\mathcal{M}}\frac{(1-\dot{\rho}_m(\boldsymbol{a}))^{1-\dot{\alpha}}-1}{\dot{\alpha}-1}\Big] \tag{44}$$

$$= \frac{1-\dot{\alpha}}{(\dot{\alpha}-1)(1-\dot{\rho}_m(\boldsymbol{a}))^{\dot{\alpha}}}\cdot\frac{\partial \dot{\rho}_m(\boldsymbol{a})}{\partial a_m^n} \tag{45}$$

$$= \frac{-1}{(1-\dot{\rho}_m(\boldsymbol{a}))^{\dot{\alpha}}}\cdot\frac{\partial \dot{\rho}_m(\boldsymbol{a})}{\partial a_m^n}. \tag{46}$$

Case 2: $\dot{\alpha}=1$,

$$\frac{\partial \mathbf{T}(\dot{\alpha},\dot{\boldsymbol{\rho}}(\boldsymbol{a}))}{\partial a_m^n} = \frac{\partial}{\partial a_m^n}\Big[-\sum_{m\in\mathcal{M}}\ln\left(\frac{1}{1-\dot{\rho}_m(\boldsymbol{a})}\right)\Big] \tag{47}$$

$$= \frac{1-\dot{\rho}_m(\boldsymbol{a})}{(1-\dot{\rho}_m(\boldsymbol{a}))^2}\cdot\frac{-\partial \dot{\rho}_m(\boldsymbol{a})}{\partial a_m^n} \tag{48}$$

$$= \frac{-1}{1-\dot{\rho}_m(\boldsymbol{a})}\cdot\frac{\partial \dot{\rho}_m(\boldsymbol{a})}{\partial a_m^n}. \tag{49}$$

Note that Eqs. (46) and (49) are equivalent when $\dot{\alpha}=1$. Thus, we obtain that

$$\frac{\partial \mathbf{T}(\dot{\alpha},\dot{\boldsymbol{\rho}}(\boldsymbol{a}))}{\partial a_m^n} = \frac{-1}{(1-\dot{\rho}_m(\boldsymbol{a}))^{\dot{\alpha}}}\cdot\frac{\partial \dot{\rho}_m(\boldsymbol{a})}{\partial a_m^n}, \qquad \forall \dot{\alpha}\geq 0. \tag{50}$$

C. Partial Derivatives of $\dot{\rho}_m(\boldsymbol{a})$

The partial derivatives $\frac{\partial \dot{\rho}_m(\boldsymbol{a})}{\partial a_m^n}$ are calculated as follows. Eq. (41) is the result of replacing $\dot{\rho}_m(\boldsymbol{a})$ with Eq. (4). By using derivative rules, we get Eq. (42) from Eq. (41). With some basic mathematical manipulation, we rewrite Eq. (42) as Eq. (43) in order to simplify the result. For the sake of clarity, we use $\widehat{IN_m^n} = \delta + \sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k$ to denote the sum of noise and inference from other classes. Recall that $\frac{a_m^n P_l^n H_m^n}{\delta+\sum_{k\in\mathcal{N}_m^n} a_m^k P_l^k H_m^k}$ is denoted by $\widehat{SINR_m^n}$. With these two variables, Eq. (43) can be rewritten as Eq. (51). Note that $\frac{W_m}{R_m^n(\boldsymbol{a})}$ in Eq. (52) can be replaced by Eq. (3), thus resulting in Eq. (53). Since $\widehat{SINR_m^n} = \frac{a_m^n P_l^n H_m^n}{\widehat{IN_m^n}}$, we can obtain Eq. (54). By substituting the function body of $g(\widehat{SINR_m^n})$ (see Eq. (14)) in Eq. (54), we get Eq. (55).

$$\frac{\partial \dot{\rho}_m(\boldsymbol{a})}{\partial a_m^n} = \frac{W_m\theta^n}{[a_m^n R_m^n(\boldsymbol{a})]^2}\left(\frac{2(a_m^n)^2 R_m^n(\boldsymbol{a})}{W_m} - \frac{(a_m^n)^2 P_l^n H_m^n}{ln(2)\widehat{IN_m^n}(\widehat{SINR_m^n}+1)}\right) \tag{51}$$

$$= \frac{2\theta^n}{R_m^n(\boldsymbol{a})} - \frac{\theta^n W_m P_l^n H_m^n}{ln(2)[R_m^n(\boldsymbol{a})]^2\widehat{IN_m^n}(\widehat{SINR_m^n}+1)} \tag{52}$$

$$= \frac{2\theta^n}{R_m^n(\boldsymbol{a})} - \frac{\theta^n a_m^n P_l^n H_m^n}{ln(2)R_m^n(\boldsymbol{a})\widehat{IN_m^n}(\widehat{SINR_m^n}+1)\log_2(\widehat{SINR_m^n}+1)} \tag{53}$$

$$= \frac{2\theta^n}{R_m^n(\boldsymbol{a})} - \frac{\theta^n \widehat{SINR_m^n}}{R_m^n(\boldsymbol{a})(\widehat{SINR_m^n}+1)\ln(\widehat{SINR_m^n}+1)} \tag{54}$$

$$= \frac{\theta^n\left(2-g(\widehat{SINR_m^n})\right)}{R_m^n(\boldsymbol{a})}. \tag{55}$$

Finally, we can obtain the partial derivatives of

$\mathbf{T}(\dot{\alpha}, \ddot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a}), \ddot{\boldsymbol{\rho}})$ as follows:

$$\frac{\partial \mathbf{T}(\dot{\alpha}, \ddot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a}), \ddot{\boldsymbol{\rho}})}{\partial a_m^n} = \xi \cdot \frac{\partial \mathbf{T}(\ddot{\alpha}, \ddot{\boldsymbol{\rho}})}{\partial a_m^n} + \frac{\partial \mathbf{T}(\dot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a}))}{\partial a_m^n} \tag{56}$$

$$= \frac{-\eta^n \xi}{F_m (1 - \ddot{\rho}_m)^{\ddot{\alpha}}} + \frac{-1}{(1 - \dot{\rho}_m(\boldsymbol{a}))^{\dot{\alpha}}} \cdot \frac{\theta^n \left(2 - g(\widehat{SINR_m^n})\right)}{R_m^n(\boldsymbol{a})} \tag{57}$$

$$= - \left[\frac{\eta^n \xi}{F_m (1 - \ddot{\rho}_m)^{\ddot{\alpha}}} + \frac{\theta^n \left(2 - g(\widehat{SINR_m^n})\right)}{R_m^n(\boldsymbol{a})(1 - \dot{\rho}_m(\boldsymbol{a}))^{\dot{\alpha}}} \right] \tag{58}$$

$$= F_m^n(\boldsymbol{a}). \tag{59}$$

According to Definition 1, The population game $\boldsymbol{F}(\boldsymbol{a}) = \{F_m^n(\boldsymbol{a}) : m \in \mathcal{M}^n, n \in \mathcal{N}\}$ is a potential game and the potential function is $\mathbf{T}(\dot{\alpha}, \ddot{\alpha}, \dot{\boldsymbol{\rho}}(\boldsymbol{a}), \ddot{\boldsymbol{\rho}})$. □