Mean Field Evolutionary Dynamics in Dense-User Multi-Access Edge Computing Systems

Hao Gao, *Graduate Student Member, IEEE*, Wuchen Li, Reginald A. Banez, Zhu Han, *Fellow, IEEE*, and H. Vincent Poor, *Life Fellow, IEEE*

***Abstract*— Multi-access edge computing (MEC) can use the distributed computing resources to serve the large numbers of mobile users in the next generation of communication systems. In this new architecture, a limited number of mobile edge servers will serve a relatively large number of mobile users. Heterogeneous servers can provide either single resource or multiple different resources to the massive number of selfish mobile users. To achieve high quality of service (QoS) and low latency under these two cases, we construct two system models and formulate our problems as two non-cooperative population games. Then we apply our proposed mean field evolutionary approach with two different strategy graphs to solve the load balancing problems under those two cases. Finally, to evaluate the performance of our algorithms, we employ the following performance indicators: overall response time (average response time of the whole system), individual response time (response time of each server), and fairness index (equality of users' response time).**

***Index Terms*— Mean field evolutionary approach, load balancing, dense-user, MEC.**

I. Introduction

AS COMBINATION of mobile computing and cloud computing, mobile cloud computing (MCC) has provided variety of services, such as computation and storage, to satisfy the demand of massive mobile users [2]. However, MCC faces significant challenges in the next generation communication networks because the total number of mobile devices continues to grow dramatically with more computation-hungry applications, such as augmented reality (AR). According to the Cisco Visual Networking Index (CVNI) report [3], there will be 12.3 billion mobile-connected devices and the average mobile network connection speed (8.7 Mbps in 2017) will reach 28.5 Megabits per second (Mbps) by 2022. It will be challenging for MCC to satisfy high Quality of Service (QoS) and low latency required by those devices with real-time applications, due to frequent uploading and downloading between users and remote core clouds [4]. As a result, new communication architectures are urgently needed.

Multi-access edge computing (MEC) is a promising candidate [5] that can handle the challenges currently being faced by MCC because of the following three characteristics. First, mobile edge hosts are deployed close to mobile devices, which means the physical distance for data transmission will be much shorter than MCC. Second, mobile edge servers can analyze big data in a local manner and run isolated from the rest of the MEC network, which means heavy uploading and downloading traffic generated by millions of mobile users will decrease sharply. Finally, MEC can provide high QoS with ultra low latency and high bandwidth. However, to turn these advantages of MEC into reality, many challenges still remain [6]. For example, in a dense-user MEC network, a limited number of servers are serving a relatively large number of users. These selfish users are only interested in their own profits such as shorter response time, while limited servers are working independently without a central controller. This will result in poor performance of the system. Therefore, an efficient load balancing algorithm is needed to achieve the good performance required by MEC [7].

A. Related Work

Many control models can be used to address the challenges of MEC, and these can be categorized into three main types: centralized control models, distributed control models and hybrid control models. Considering the large system scale and decentralized manner of dense-user MEC systems, our approach falls into the category of distributed control models. There are many distributed control models in the recent literature which can be categorized into the following four classes [8]:

1) Distributed Optimization Methods: Reference [9] provided a fully distributed algorithm considering the load balancing problem as a global optimization problem. In this distributed algorithm, the whole system is modeled as a graph where the nodes are the users and they are required to know which constraint they are involved with. Nevertheless, this algorithm is only suitable for a convex separable payoff function over linear inequality constraints.

2) Greedy Heuristics Techniques: In these approaches, the load balancing problems are modeled as a non-cooperative

Manuscript received October 31, 2019; revised March 2, 2020, June 11, 2020, and July 31, 2020; accepted August 7, 2020. Date of publication August 21, 2020; date of current version December 10, 2020. This work was supported by the U.S. Air Force Office of Scientific Research under MURI Grant FA9550-18-1-0502. This article was presented at the 2019 IEEE Global Communication Conference [1]. The associate editor coordinating the review of this article and approving it for publication was L. Duan. *(Corresponding author: Hao Gao.)*

Hao Gao, Reginald A. Banez, and Zhu Han are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204 USA (e-mail: hgao5@uh.edu; rabanez@uh.edu; hanzhu22@gmail.com).

Wuchen Li is with the University of South Carolina, Columbia, SC 29225 USA (e-mail: wuchen@mailbox.sc.edu).

H. Vincent Poor is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: poor@princeton.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2020.3016695

1536-1276 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Houston. Downloaded on February 08,2021 at 01:41:05 UTC from IEEE Xplore. Restrictions apply.

game. Reference [10] formulated the problem as a non-cooperative game and adopted the Nash Equilibrium as the user-optimal solution. Reference [11] also regarded the load balancing problem in the data center as a non-cooperative game, and constructed a decentralized and low-complexity load balancing algorithm for computing the Nash equilibrium. However, these approaches are mainly based on greedy heuristics and they can not guarantee the optimality of the solution [8].

3) Leader-Follower Based Approaches: References [12] and [13] formulated the load balancing problem as an optimization problem via a Stackelberg game, where one player is regarded as the leader (centralized authority intrigued in optimizing the performance of the whole system) and the rest of players are viewed as the followers (the self-interested players). In the leader-follower model, the leader has the ability to influence the task allocation decisions. Nevertheless, these approaches are only suitable for non-linear payoff functions [8].

4) Population Based Approaches: Reference [8] formulated the load balancing problem in distributed systems as a non-cooperative population game and viewed the Nash equilibrium as the user-optimal solution to the game. The authors achieved an algorithm that has a distributed nature, user optimality, near-optimal solution, and fair allocation. However, their analysis of the existence of the Nash equilibrium are mainly inferred from the simulation results rather than proof by analysis.

Besides the disadvantages just mentioned, these four categories of methods have two common weaknesses if we apply them to dense-user MEC systems: (i) they do not consider a large scale system with numbers of heterogeneous users; (ii) and they do not consider the case in which servers can provide multiple different resources.

B. Motivation and Contribution

In this paper, we apply a mean field evolutionary approach to solve the load balancing problem in dense-user MEC for the following reasons: (i) Separately dealing with the interactions between a large number of users will increase the computational complexity significantly for other game theoretic methods, such as the Nash non-cooperative game and traditional evolutionary game [14], [15]. (ii) Instead of reacting to other users separately when seeking the optimal strategy for a generic user, a mean field game (MFG) regards them as a mean field and reacts only to the collective behavior [16]. This will reduce the computational complexity significantly [17]. (iii) We consider both the single-resource case and the multi-resource case of the servers. The single-resource case is the situation when the server can only provide one type of resource to the user, such as computation. The multi-resource case refers to the situation when the server can provide multiple types of resources to the user, such as computation and storage. By designing different strategy graphs, our approach can solve the problem not only in the single-resource case but also in the multi-resource case. The main contributions of this paper are as follows:

- We construct two dense-user MEC models for the single-resource case and multi-resource case.
- We apply a novel mean field evolutionary mechanism and design two different strategy graphs to solve the problems under those two different cases.
- We achieve a near-optimal solution in the single-resource case, and also improve the performance of the whole system significantly in the multi-resource case.

The rest of this paper is organized in the following way. In Section II, we model the dense-user MEC network in both the single-resource case and multi-resource case. Moreover, we formulate the load balancing problems in those cases as non-cooperative dynamic population games. In Sections III and IV, effective mean field evolutionary approaches, with one-dimensional and two-dimensional strategy graphs, are proposed to solve the problems under the two cases, respectively. In Section V, we conduct a comprehensive simulation and Section VI concludes the paper.

II. Modeling and Formulating

We show our general system model in Fig. 1. The mobile server, which is a small-box data center consisting of several multi-core computers [2], are considered as the servers in the dense-user MEC systems. They are working independently without a central controller and are denoted as $\mathcal{S} = \{1, 2, \cdots, s\}$. Furthermore, those servers may provide either single resource or multiple different resources to the mobile users. On the other hand, users denoted by $\mathcal{U} = \{1, 2, \cdots, u\}$ are only interested in their own profit. The jobs of all users are assumed to be homogeneous and indistinguishable without priorities, deadlines and multiple versions [18], because heterogeneous workload needs to be divided into different populations, which falls into the framework of multi-population mean field game [19], [20]. After defining server set $\mathcal{S}$ and user set $\mathcal{U}$, we can construct the system models in both single-resource case and multi-resource case as follows.

A. Single-Resource Case

User i's jobs ($i \in \mathcal{U}$) are generated according to a Poisson process with mean rate λ_i, and $\hat{\lambda} = \sum_{i=1}^{u} \lambda_i$ denotes the total arrival rate of all users. User i will seek its own best strategy to allocate its jobs to all servers. Server j ($j \in \mathcal{S}$) is modeled by $M/M/1$ [21] with a service rate μ_j. To keep a stable system, we need to ensure that $\hat{\lambda} < \sum_{j=1}^{s} \mu_j$ and the system utilization rate is $\theta = \frac{\hat{\lambda}}{\sum_{j=1}^{s} \mu_j}$.

User i's strategy is denoted as $\chi_i \in \mathbb{R}^s$. χ_{ij} represents the proportion of user i's total amount of jobs assigned to server j. Therefore, user i's strategy can be considered as a mixed strategy on the server set $\mathcal{S}$. We record all users' strategies in one strategy profile $\mathcal{X} = (\chi_i)_{i=1}^{u}, \mathcal{X} \in \mathbb{R}^{u \times s}$ and each row χ_i is the mixed strategy of user i. The distribution of users' jobs on the servers is the "mean field".

When users determine their strategies, they want to know their individual response time. In order to obtain the expected response time of user i, we should know the response time of server j to user i first, because user i's jobs are assigned on

Authorized licensed use limited to: University of Houston. Downloaded on February 08,2021 at 01:41:05 UTC from IEEE Xplore. Restrictions apply.

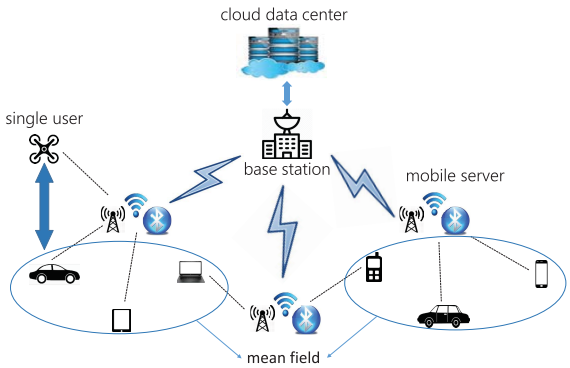


Fig. 1. System model.

different servers. Under the single-resource case, the response time of server j is given by:

$$R_j(\mathcal{X}) = \frac{1}{\mu_j - \sum_{n=1}^{u} \lambda_n \chi_{nj}}, \tag{1}$$

where μ_j is server j's service rate and $\sum_{n=1}^{u} \lambda_n \chi_{nj}$ is total workload on server j. Then the total expected response time is given by:

$$\begin{aligned} T_i(\mathcal{X}) &= \sum_{j=1}^{s} \chi_{ij} R_j(\mathcal{X}) \\ &= \sum_{j=1}^{s} \frac{\chi_{ij}}{\mu_j - \sum_{n=1}^{u} \lambda_n \chi_{nj}}, \end{aligned} \tag{2}$$

where $\chi_{ij} R_j(\mathcal{X})$ is the response time of server j to user i weighted by user i's probability of selecting server j.

When we try to find the optimal strategies for users to minimize their expected response time, we need to satisfy the following constraints according to [10], [22], [23], and [24]:

$$\chi_{ij} \geqslant 0, \quad \forall i \in \mathcal{U}, \ \forall j \in \mathcal{S}, \tag{3}$$

$$\sum_{j=1}^{s} \chi_{ij} = 1, \quad \forall i \in \mathcal{U}, \tag{4}$$

$$\sum_{n=1}^{u} \lambda_n \chi_{nj} \leqslant \mu_j, \quad \forall j \in \mathcal{S}. \tag{5}$$

(3) is recommended because χ_{ij} is the percentage of user i's jobs assigned to server j. (4) needs to be satisfied because user i will allocate his jobs on all or some of the servers. In (5), $\sum_{n=1}^{u} \lambda_n \chi_{nj}$ is the workload on server j. Therefore, we need to keep the workload less than the service rate μ_j to ensure the stability of the system.

We regard the Nash Equilibrium [25], where all users have no incentive to modify their strategies unilaterally, as the solution to this non-cooperative population game. When we seek the optimal strategies to achieve the Nash Equilibrium, we will update the strategy of user i while keeping strategies of other users fixed with our mean field evolutionary methodology. Considering this, solving the non-cooperative population game for multi-users is equivalent to solving the following optimization problem for a single user:

$$\begin{aligned} \min_{\mathcal{X}} \ & T_i(\mathcal{X}) = \sum_{\mathbf{j=1}}^{\mathbf{s}} \frac{\chi_{\mathbf{ij}}}{\mu_{\mathbf{j}} - \sum_{\mathbf{n=1}}^{\mathbf{u}} \lambda_{\mathbf{n}} \chi_{\mathbf{nj}}}, \\ \text{s.t. } & \chi_{ij} \geq 0, \quad \forall j \in \mathcal{S}, \\ & \sum_{j=1}^{s} \chi_{ij} = 1, \\ & \sum_{n=1}^{u} \lambda_n \chi_{nj} < \mu_j, \quad \forall j \in \mathcal{S}, \end{aligned} \tag{6}$$

where $T_i(\mathcal{X})$ is defined in (2) and the three constraints are (3), (4), and (5), respectively.

B. Multi-Resource Case

For the multi-resource case, each server provides multiple different resources for users to complete corresponding types of jobs. Therefore, aside from user set $\mathcal{U}$ and server set $\mathcal{S}$, we need to define the resource set as $\mathcal{R} = \{1, 2, \cdots, r\}$.

User i is generating the jobs of class i requiring r different resources according to a Poisson process with mean rate $\lambda_i = (\lambda_{i1}, \lambda_{i2}, \cdots, \lambda_{ir}), \lambda_i \in \mathbb{R}^r$. The kth entry of λ_i represented the rate of generating the kth type of jobs requiring the kth type of resources. The job profile $\lambda = (\lambda_i)_{i=1}^{u}$ records all users' job generating rates. These jobs are assigned to all the servers and each server is modeled by $M/M/1$ [21]. The service rate of server j is $\mu_j = (\mu_{jk})_{k=1}^{r} \in \mathbb{R}^k$, with the kth entry denoting service rate of the kth type of jobs. Thus $\mu = (\mu_j)_{j=1}^{s}$ is the $s \times r$ matrix recording all servers' service rate.

Similar to the single-resource case, each user will allocate its different types of jobs to the servers. However, under the multi-resource case, a generic user i not only needs to determine which server to connect but also which type of resource to utilize. Therefore, the strategy set for multi-resource case is the $r \times s$ different combinations of server set $\mathcal{S}$ and resource set $\mathcal{R}$, which is defined as follows:

$$\hat{\mathcal{S}} = \{(j, k) : j \in \mathcal{S}, k \in \mathcal{R}\}.$$

Denoting the strategy of user i as $\pi_i \in \mathbb{R}^t, t = r \times s$, it is the probability distribution on $\hat{\mathcal{S}}$. The $j + s(k-1)$ entry represents the probability of selecting k type of resource on server j. π_i is recorded in this way because it is convenient for us to update π_i later with gradient matrix and our strategy graph is a 2-D lattice as shown in Fig. 2. Hence, $\pi = (\pi_i)_{i=1}^{u}$ is the strategy profile recording all users' strategies.

As user i's jobs are assigned to all the servers, we need to compute the response time of each server j in order to obtain user i's total response time. Moreover, the response time of server j should be the total response time of all types of jobs. Therefore, the derived response time of server j to user i will be:

$$R_j(\pi) = \sum_{k=1}^{r} \frac{1}{\mu_{jk} - \sum_{n=1}^{u} \lambda_{nk} \pi_{nm}}, \tag{7}$$

where $m = j + s(k-1)$ with $j \in \mathcal{S}, k \in \mathcal{R}$ and $\sum_{n=1}^{u} \lambda_{nk} \pi_{nm}$ is the total kth type of jobs assigned on server j. Consequently,

Authorized licensed use limited to: University of Houston. Downloaded on February 08,2021 at 01:41:05 UTC from IEEE Xplore. Restrictions apply.

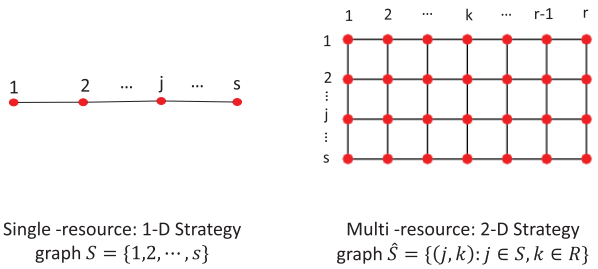


Fig. 2. Strategy graphs for the single-resource case and the multi-resource case.

the total expected response time of user i is computed by:

$$T_i(\pi) = \sum_{j=1}^{s} \pi_{im} R_j(\pi) = \sum_{j=1}^{s}\sum_{k=1}^{r} \frac{\pi_{im}}{\mu_{jk} - \sum_{n=1}^{u} \lambda_{nk}\pi_{nm}}, \quad (8)$$

where $\pi_{im} R_j(\pi)$ is server j's response time weighted by the probability of selecting resource k on server j.

Similar to the single-resource case, certain preferences are recommended by [10], [22], [23], and [24]. They are given as follows:

$$\pi_{im} \geqslant 0, \quad (9)$$

$$\sum_{j=1}^{s}\sum_{k=1}^{r} \pi_{im} = 1, \quad (10)$$

$$\sum_{n=1}^{u} \lambda_{nk}\pi_{nm} < \mu_{jk}, \quad (11)$$

where $m = j + s(k-1), i \in \mathcal{U}, j \in \mathcal{S}, k \in \mathcal{R}$. The first and second are suggested because π_{im} is probability distribution on $\hat{\mathcal{S}}$. The third needs to be satisfied for stability of the whole system.

With the system constraints defined above and the expected response time $T_i(\pi)$ as the objective function, we still formulate our problem for the multi-resource case as the following optimization problem for a generic user i:

$$\begin{aligned} \min_{\pi} \ & \sum_{j=1}^{s}\sum_{k=1}^{r} \frac{\pi_{im}}{\mu_{jk} - \sum_{n=1}^{u} \lambda_{nk}\pi_{nm}}, \\ \text{s.t. } & \pi_{im} \geqslant 0, \\ & \sum_{j=1}^{s}\sum_{k=1}^{r} \pi_{im} = 1, \\ & \sum_{n=1}^{u} \lambda_{nk}\pi_{nm} < \mu_{jk}, \end{aligned} \quad (12)$$

where $m = j + s(k-1)$, $j \in \mathcal{S}, k \in \mathcal{R}$. The objective function in (12) is the the total expected response time defined in (8) and the three constraints are (9), (10), and (11), respectively.

III. One Dimensional Mean Field Evolutionary Approach

In this section, we propose the one dimensional (refer to the dimension of strategy graph as shown in Fig. 2) mean field evolutionary approach for the single-resource case. The core steps of our approach are manifested in Algorithm 1. Before the detailed analysis of each step in the next three subsections, we must mention that (3), (4), and (5) are actually satisfied after each iteration. Constraints (3) and (4) are satisfied because our mean field evolutionary dynamic is defined on the probability space given by (14). As for constraint (5), we only need to keep the following inequality (13) satisfied during the iterations:

$$\max\left(\sum_{n=1}^{u} \lambda_n \chi_{nj}\right) < \mu_j. \quad (13)$$

Therefore, (5) is satisfied because we can ensure (13) after initialization and $\max(\sum_{i=1}^{u} \lambda_i \chi_{ij})$ always decreases as we observe in the simulation, while the service rate μ_j is a constant.

Algorithm 1 1-D Mean Field Evolutionary Approach for Single-Resource Case

```
random initialization(𝒳, λ, μ)
while (25) is not satisfied do
    for each user i do
        gradient matrix Λ_i construction by (21), (22) and (23)
        update user i's strategy 𝒳_i by (24)
    end for
end while
Return: strategy profile 𝒳
```

A. Gradient Matrix Construction

In this subsection, we utilize MFG to construct the Riemannian manifold and then derive the mean field evolutionary dynamics (Theorem 1) on the Riemannian manifold. Finally, a gradient matrix $\Lambda \in \mathbb{R}^{s\times s}$ is constructed based on Theorem 1.

1) Riemannian Manifold Construction With MFG: We consider a MFG on graph $\mathcal{G} = (\mathbb{S}, \mathbb{E})$:

- **Nodes and Edges:** Nodes of this graph are pure strategies from the discrete strategy set $\mathbb{S} = \{1, 2, \cdots, \alpha\}$. Edges are connections between nodes. Node $i \in \mathbb{S}$ and node $j \in \mathbb{S}$ are able to form an edge $(i, j) \in \mathbb{E}$ if players can directly switch from strategy i to strategy j.
- **Neighborhood:** The neighborhood of node i is the set of all nodes which have a direct connection to node i. It's defined as follows:

$$N(i) = \{j \in \mathbb{S} : (i, j) \in \mathbb{E}\}.$$

- **State Space:** The state space consists of all available mixed strategies of the players and it is defined in the following way:

$$\mathcal{P}(\mathbb{S}) = \{(\rho_i)_{i=1}^{\alpha} : \sum_{i=1}^{\alpha} \rho_i = 1, \rho_i \geq 0, i \in \mathbb{S}\}, \quad (14)$$

where ρ_i represents the probability of selecting strategy i. The interior of $\mathcal{P}(\mathbb{S})$ is denoted as $\mathcal{P}_o(\mathbb{S})$.

In order to measure the distance in the state space, we need to define the Wasserstein metric.

Authorized licensed use limited to: University of Houston. Downloaded on February 08,2021 at 01:41:05 UTC from IEEE Xplore. Restrictions apply.

Definition 1: Given two discrete probability functions $\rho^0, \rho^1 \in \mathcal{P}_o(\mathbb{S})$, the Wasserstein metric W is defined by:

$$W(\rho^0,\rho^1)^2 = \inf\left\{\int_0^1 (\nabla\Phi(t),\nabla\Phi(t)_{\rho(t)})dt : \frac{d\rho}{dt} + div(\rho\nabla\Phi) = 0, \rho(0)=\rho^0, \rho(1)=\rho^1\right\},$$

where $\nabla\Phi : \mathbb{S}\times\mathbb{S}\to\mathbb{R}$ is given by:

$$\nabla\Phi = \begin{cases}\Phi_i - \Phi_j, & \text{if } (i,j)\in\mathbb{E},\\ 0, & \text{otherwise},\end{cases}$$

where Φ is a function and $\Phi : \mathbb{S}\to\mathbb{R}$.

Besides the Wasserstein metric, we also need the following inner product g^W to construct the Remannian manifold $(\mathcal{P}_o(\mathbb{S}), g^W)$.

Definition 2: For any two tangent vectors $\sigma^1,\sigma^2 \in T_\rho\mathcal{P}_o(\mathbb{S})$, define the inner product $g^W : T_\rho\mathcal{P}_o(\mathbb{S})\times T_\rho\mathcal{P}_o(\mathbb{S})\to\mathbb{R}$ by:

$$g^W(\sigma^1,\sigma^2) = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}}\theta_{ij}(\rho)(\Phi_i^1-\Phi_j^1)(\Phi_i^2-\Phi_j^2), \quad (15)$$

where $\sigma^i = -div(\rho\nabla\Phi^i)$ for $i=1,2$. $T_\rho\mathcal{P}_o(\mathbb{S}) = \{(\sigma_i)_{i=1}^n \in \mathbb{R}^n : \sum_{i=1}^n \sigma_i = 0\}$ is the tangent space at a point $\rho\in\mathcal{P}_o(\mathbb{S})$. θ_{ij} is the discrete probability on edge (i,j) which is defined by:

$$\theta_{ij}(\rho) = \begin{cases}\frac{1}{d_j}\rho_j & F_j(\rho) < F_i(\rho),\\ \frac{1}{d_i}\rho_i & F_j(\rho) > F_i(\rho),\\ \frac{1}{2}(\frac{\rho_i}{d_i}+\frac{\rho_j}{d_j}) & F_j(\rho) = F_i(\rho),\end{cases} \quad (16)$$

where $F_i : \mathcal{P}(\mathbb{S})\to\mathbb{R}$ is the payoff function and d_i is the degree of node i (i.e. the total number of nodes in $N(i)$).

With the state space in (14) and the inner product in Definition 2, we finally construct the Riemannian manifold $(\mathcal{P}_o(\mathbb{S}), g^W)$ [26] on which the mean field evolutionary dynamic can be defined.

2) Fokker-Planck Equation as Evolutionary Dynamic: In this subsection, we define the mean field evolutionary dynamic with the Fokker-Planck equation on the Riemannian manifold $(\mathcal{P}_o(\mathbb{S}), g^W)$. The mean field evolutionary dynamic is given by the Theorem 1.

Theorem 1: Suppose that a population game has strategy graph $\mathcal{G} = (\mathbb{S},\mathbb{E})$ and the constant $\beta\geq 0$, the utility function $F : \mathcal{P}(\mathbb{S})\to\mathbb{R}^n$ are continuous. Then for any initial condition $\rho^0\in\mathcal{P}_o(\mathbb{S})$, the Fokker-Planck equation

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = \sum_{j\in N(i)}\frac{1}{d_j}\rho_j[F_i(\rho)-F_j(\rho)+\beta(\log\rho_j-\log\rho_i)]^+ - \sum_{j\in N(i)}\frac{1}{d_i}\rho_i[F_j(\rho)-F_i(\rho)+\beta(\log\rho_i-\log\rho_j)]^+ \quad (17)$$

is a well-defined gradient flow in $\mathcal{P}_o(\mathbb{S})$. β represents the strength of uncertainty, $[\cdot]^+ = max\{\cdot,0\}$, $N(i)$ is the neighborhood of node i, and the degree of node j is denoted as $d_j = \sum_{i\in N(j)} 1$.

Remark 1: In (17), the degree of node j, d_j and the neighborhood of node i, $N(i)$ are determined by the structure of the strategy graph. For instance, as shown in the 1-D strategy graph in Fig. 2, $d_i = 2, \forall i\neq 1,s$ while $d_1 = d_s = 1$, and $N(i) = \{i-1, i+1\}, \forall i\neq 1,s$ while $N(1) = \{2\}, N(s) = \{s-1\}$. Through d_j and $N(i)$, the strategy graph can determine how users' strategies are updated during each iteration and also the final convergence of the mean field evolutionary dynamics.

Proof: Given the tangent space $T_\rho\mathcal{P}_o(\mathbb{S}) = \{(\sigma_i)_{i=1}^n \in \mathbb{R}^n : \sum_{i=1}^n\sigma_i = 0\}$, there exists Φ such that $\sigma = -div(\rho\nabla\Phi)$ for any $\sigma\in T_\rho\mathcal{P}_o(\mathbb{S})$. As $\frac{d\rho}{dt} = (\frac{d\rho_i}{dt})_i^n$ is in $T_\rho\mathcal{P}_o(\mathbb{S})$, we have

$$g^W\left(\frac{d\rho}{dt},\sigma\right) = \sum_{i=1}^n\frac{d\rho_i}{dt}\Phi_i, \quad (18)$$

The noisy potential is given by:

$$\bar{F}(\rho) = F(\rho) - \beta\sum_{i=1}^n\rho_i\log\rho_i, \quad \beta\geq 0, \quad (19)$$

which is the summation of the potential and the Shannon-Boltzmann entropy. Then we have

$$d\bar{F}(\rho)\cdot\sigma = \sum_{i=1}^n\frac{\partial}{\partial\rho_i}\bar{F}(\rho)\cdot\sigma_i = -\sum_{i=1}^n\bar{F}_i(\rho)div(\rho\nabla\Phi)_i = (\nabla\bar{F}(\rho),\nabla\Phi) = -\sum_{i=1}^n\Phi_i div(\rho\nabla\bar{F}(\rho))_i. \quad (20)$$

With (18) and (20), and the definition of gradient flow of $-\bar{F}(\rho)$ on the Riemannian manifold $(\mathcal{P}_o(\mathbb{S}), g^W)$, we derive

$$0 = g^W(\frac{d\rho}{dt},\sigma) - d\bar{F}(\rho)\cdot\sigma = \sum_{i=1}^n\frac{d\rho_i}{dt} + div(\rho\nabla\bar{F}(\rho))_i\Phi_i.$$

As the above is true for all $(\Phi_i)_{i=1}^n\in\mathbb{R}^n$, we finally obtain

$$\frac{d\rho_i}{dt} + \sum_{j\in N(i)}\theta_{ij}(\rho)(\bar{F}_j(\rho)-\bar{F}_i(\rho)) = 0.$$

Replacing θ_{ij} with (16), the mean field evolutionary dynamic in Theorem 1 is proved. □

The mean field evolutionary dynamic in Theorem 1 is a gradient flow starting from any initial ρ^o in the state space $\mathcal{P}_o(\mathbb{S})$ to the optimal $\rho^\star$, which can optimize the continuous utility function F. Therefore, when we utilize Theorem 1 to solve the optimization problem given in (6), every strategy $\mathcal{X}_i$ is a point in the state space and the total expected response time will be regarded as the continuous utility function. Moreover, we need to construct a gradient matrix Λ with Theorem 1 to update the strategy profile $\mathcal{X}$ in an iterative manner.

3) Gradient Matrix Computation: In this subsection, we will utilize Theorem 1 to construct the gradient matrix $\Lambda\in\mathbb{R}^{s\times s}$. First, we need to define the noisy potential $F(j)$ as required in Theorem 1. $F(j), j\in\mathcal{S}$ is the first-order derivative

Authorized licensed use limited to: University of Houston. Downloaded on February 08,2021 at 01:41:05 UTC from IEEE Xplore. Restrictions apply.

of user i's expected response time $T_i(\mathcal{X})$, which is computed by:

$$F(j) = \frac{\partial}{\partial \mathcal{X}_{ij}} T_i(\mathcal{X}) = \frac{\mu_j - \sum_{n=1}^{u} \lambda_n \mathcal{X}_{nj} + \lambda_i \mathcal{X}_{ij}}{(\mu_j - \sum_{n=1}^{u} \lambda_n \mathcal{X}_{nj})^2}, \quad \forall j \in \mathcal{S}. \quad (21)$$

As shown in the 1-D strategy graph in Fig. 2, server 1 and server s only have connection to server 2 and server $s-1$, respectively, their gradients will be computed by ($\beta = 0$):

$$\begin{aligned} \Lambda(1,1) &= -[F(1) - F(2)]^+, \\ \Lambda(1,2) &= [F(2) - F(1)]^+, \\ \Lambda(s,s) &= -[F(s) - F(s-1)]^+, \\ \Lambda(s,s-1) &= [F(s-1) - F(s)]^+. \end{aligned} \quad (22)$$

Denoting the set of all other servers in the MEC systems as $\mathcal{S}^o = \{i \in \mathcal{S} : i \neq 1, s\}$, for any $i \in S^o$, they have connections to server $i-1$ and server $i+1$. Therefore, their gradients should be computed by ($\beta = 0$):

$$\begin{aligned} \Lambda(i,i) &= -[F(i) - F(i+1)]^+ - [F(i) - F(i-1)]^+, \\ \Lambda(i,i+1) &= [F(i+1) - F(i)]^+, \quad \forall i \in \mathcal{S}^o, \\ \Lambda(i,i-1) &= [F(i-1) - F(i)]^+. \end{aligned} \quad (23)$$

In (22) and (23), $F(i), i \in \mathcal{S}$, is the noisy potential function given in (21). Apart from the entries specified in (22) and (23), all the other entries of the gradient matrix $\Lambda \in \mathbb{R}^{s \times s}$ are set to 0.

B. Update Strategy

We update user i's strategy $\mathcal{X}_i$ in an iterative manner with its gradient matrix Λ_i constructed after each iteration. At the beginning of iteration a, the strategy profile $\mathcal{X}^{a-1}$ from iteration $a-1$ is preserved to compute user i's gradient matrix Λ_i^a. After constructing Λ_i^a based on (22) and (23), user i's new strategy $\mathcal{X}_i^a$ after iteration a will be obtained through

$$\mathcal{X}_i^a = \mathcal{X}_i^{a-1} + \Lambda_i^a \mathcal{X}_i^{a-1} dt, \quad (24)$$

where dt is an appropriate time step. This whole updating process will be stopped when the following condition is satisfied:

$$\sum_{i=1}^{u} \mid \mathcal{X}_i^a - \mathcal{X}_i^{a-1} \mid \leqslant \eta, \quad (25)$$

where η is the threshold we give at the beginning of the updating process.

C. A Numerical Toy Example

Assume that 2 servers are serving 5 users. The step length of updating is set as $dt = 0.01$. As shown in Algorithm 1, we first initialize the strategy profile as

$$\mathcal{X} = \begin{bmatrix} 0.4807 & 0.5403 & 0.4192 & 0.3905 & 0.4913 \\ 0.5193 & 0.4597 & 0.5808 & 0.6095 & 0.5087 \end{bmatrix}^T,$$

Algorithm 2 2-D Mean Field Evolutionary Dynamics Mechanism for the Multi-Resource Case

```
random initialization(π, λ, μ)
while (28) is not satisfied do
    for every user in the system do
        gradient matrix construction
        update all strategies simultaneously by (27)
    end for
end while
Return: strategy profile π
```

the service rate $\mu = [12.5732\ 14.7776]$, and jobs of arrival rate

$$\lambda = [5.8104\ 3.5951\ 7.9981\ 5.2756\ 1.8264].$$

Each row of $\mathcal{X}$ represents the workload assignment strategy for the corresponding user. Then we construct the gradient matrix for each user and update their strategies based on Theorem 1. For example, the gradient matrix for the 5th user is given by

$$\Lambda_5 = \begin{bmatrix} -3.3048 & 0 \\ 3.3048 & 0 \end{bmatrix}$$

Then the new strategy for 5th user is computed in the following way:

$$\Lambda_5 \times \mathcal{X}_5 \times dt + \mathcal{X}_5 = \begin{bmatrix} -3.3048 & 0 \\ 3.3048 & 0 \end{bmatrix} \begin{bmatrix} 0.4913 \\ 0.5087 \end{bmatrix} \times 0.01 + \begin{bmatrix} 0.4913 \\ 0.5087 \end{bmatrix}$$

Hence the new strategy for user 5 is $\mathcal{X}_5 = [0.4750\ 0.5250]$. All users' strategies will be updated in the same way until the terminal condition is satisfied.

IV. Two Dimensional Mean Field Evolutionary Approach

In this section, we address the multi-resource case with a two dimensional (refer to the dimension of strategy graph as shown in Fig. 2) mean field evolutionary approach. The core procedures of our approach will be given in Algorithm 2 and detailed analysis of each step will be provided in the next two subsections.

A. Gradient Matrix Construction

As the MFG motivation and the proof of the mean field evolutionary dynamic has been given under the single-resource case, we will directly apply Theorem 1 into the multi-resource case. Following the same procedure of constructing gradient matrix in the single-resource case, we give the potential function for $T_i(\pi)$ as follows:

$$F(j,k) = \frac{\partial}{\partial \pi_{im}} T_i(\pi), = \frac{\mu_{jk} - \sum_{n=1}^{u} \lambda_{nk} \pi_{nm} + \lambda_{ik} \pi_{im}}{(\mu_{jk} - \sum_{n=1}^{u} \lambda_{nk} \pi_{nm})^2}, \quad (26)$$

where $m = j + s(k-1)$, $j \in \mathcal{S}, k \in \mathcal{R}$.

Authorized licensed use limited to: University of Houston. Downloaded on February 08,2021 at 01:41:05 UTC from IEEE Xplore. Restrictions apply.

With potential function $F(j,k)$ and Theorem 1, we can construct the gradient matrix Λ based on the two dimension strategy graph in Fig. 2. The nodes in the lattice represent pure strategies in $\hat{\mathcal{S}}$. All strategies in the kth column have chosen resource k and all strategies in the jth row have selected server j. In order to assign values through (17), the nodes on the lattice in Fig. 2 need to be classified into three different categories (vertices, edge points, and interior points) according to the number of their adjacent nodes. Each of the four vertex has two adjacent nodes and each edge point has three adjacent nodes. As for the interior points on the lattice, they have direct connections to four nodes.

B. Update Strategy

In order to reduce the computation complexity, we still update all users' strategies simultaneously. Therefore, during iteration a, strategy profile π^{a-1} from previous iteration will be preserved for the computation of user i's gradient matrix Λ_i^a. Updating rule of user i's strategy at iteration a is given by:

$$\pi_i^a = \pi_i^{a-1} + \Lambda_i^a \pi_i^{a-1} dt, \tag{27}$$

where dt is an appropriate step size. The terminal condition of this updating process is given by:

$$\sum_{i=1}^{u} \mid \pi_i^a - \pi_i^{a-1} \mid \leqslant \eta, \tag{28}$$

where η is the threshold we give at the beginning of the updating process.

V. Simulation Result

To evaluate the effectiveness of our mean field evolutionary algorithm, we conduct a comprehensive experiment in MATLAB R2019a and utilize the following three performance indicators which are frequently used in the existing literature such as [8], [10], and [22].

- Individual response time: we randomly select several users and compute their individual response time by (2) (single-resource) and (8) (multi-resource)
- Overall response time: a manifestation of the whole MEC system which is defined by the following equation.

$$\hat{T} = \frac{1}{\hat{\lambda}} \sum_{i=1}^{u} \lambda_i T_i, \tag{29}$$

  where $\hat{\lambda}_i$ is the total amount of jobs and λ_i is the entire amount of user i's jobs.
- Fairness index: an indicator of equality between users' individual response time which is computed by:

$$\mathcal{F}(T) = \frac{(\sum_{i=1}^{u} T_i)^2}{u \sum_{i=1}^{u} T_i^2}, \tag{30}$$

  where $T = (T)_{i=1}^{u}$ records all users' response time.

TABLE I
Simulation Parameters

Parameter	Single Resource	Multiple Resources
Number of servers	10	10
Number of users	100	100
Number of resources	1	5
Job arriving rates	$\mid \mathcal{N}(5,2) \mid$	$\mid \mathcal{N}(5,1) \mid$
Users' strategy profile	Normalize $\mid \mathcal{N}(5,1) \mid$	Normalize $\mid \mathcal{N}(4,2) \mid$
System utilization rate	90%	90%
Servers' service rates	Computed by (31)	Computed by (32)

A. Parameters Initialization

The basic simulation parameters for both single-resource case and multi-resource case are given in Table I. We assume that there are 10 mobile edge hosts serving 100 mobile users. Users' jobs arriving rate are initialized as the absolute value of a Gaussian random variable as shown in Table I. Then we initialize each entry of the strategy profile as the absolute value of a Gaussian random variable and normalize the entries row by row to ensure the constraints (3), (4), (9), and (10). The service rate under the single-resource case is computed by:

$$\mu_j = \frac{\sum_{n=1}^{u} \lambda_n \chi_{nj}}{\theta_j}, \quad \forall j \in \mathcal{S}. \tag{31}$$

where θ_j is the utilization rate of server j. The service rate under the multi-resource case is computed by:

$$\mu_{jk} = \frac{\sum_{n=1}^{u} \lambda_{nk} \pi_{nm}}{\theta_{jk}}, \quad \forall j \in \mathcal{S}, \ \forall k \in \mathcal{R}, \tag{32}$$

where $m = j + s(k-1)$ and θ_{jk} is the utilization rate of k type of resource on server j.

B. Performance Analysis for Single-Resource Case

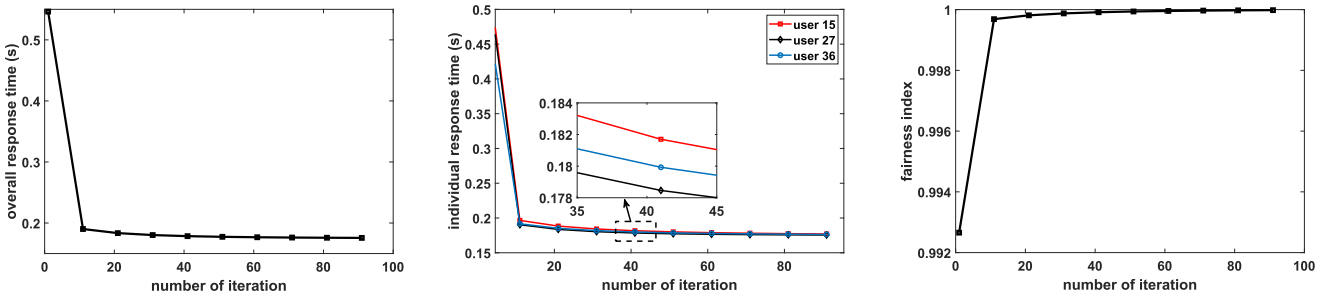
In this subsection, we evaluate the performance of our 1-D mean field evolutionary approach in the single-resource case. Convergence analysis is given firstly. Then we evaluate the efficiency of load balance. Finally, we compare our approach with [27] and [24].

1) Convergence Analysis: In Fig. 3, convergence patterns of three performance indicators have been depicted. Specifically, Fig. 3a shows the convergence behavior of the overall response time. Fig. 3b shows the individual response time of three randomly selected users. Before the adjustment of workload assignment, the individual response time among users varies from each other. After the reassigning the workload, the individual response time turns out to be close to each other. In Fig. 3c, convergence patterns of the fairness index are depicted.

2) Efficiency of Load Balance: When $\theta = 90\%$ and the computing resources are quite limited, load disproportion will influence the performance of the whole system significantly as shown in Fig. 4. Fully utilizing the idle computing resources from some light load users by reallocating users' jobs is guaranteed to improve the performance of the whole MEC systems [28].

As shown in Fig. 4, the jobs are randomly assigned to the servers at first. Some servers receive a heavy workload such as server No. 5 and No. 8 with only 1 percentage of total capacity

Authorized licensed use limited to: University of Houston. Downloaded on February 08,2021 at 01:41:05 UTC from IEEE Xplore. Restrictions apply.



(a) Overall response time (b) Individual response time (c) Fairness index

Fig. 3. Convergence analysis of single-resource case.

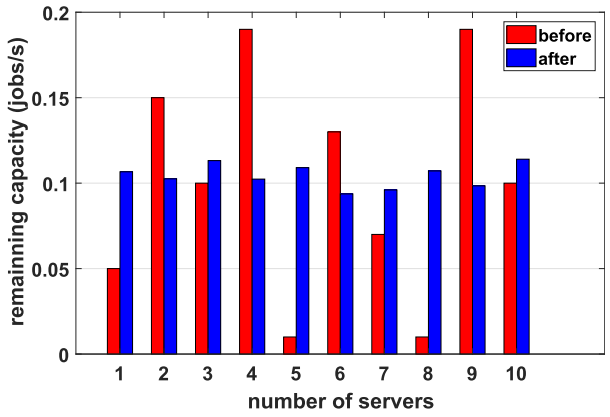


Fig. 4. Effect of load balance ($\theta = 90\%$).

TABLE II
COMPARISON ON UTILIZATION RATE OF SERVERS

Server number	Service rate(jobs/s)	Utilization rate of servers (%) with $\theta = 89.59\%$		
		MFEA	GOS	PROP
1	54.95	89.34	89.23	89.59
2	61.08	89.75	89.78	89.59
3	57.91	88.68	89.50	89.59
4	64.19	89.76	90.03	89.59
5	53.73	89.10	89.10	89.59
6	59.31	90.63	89.63	89.59
7	57.27	90.40	89.45	89.59
8	53.75	89.29	89.11	89.59
9	66.20	90.16	90.18	89.59
10	59.19	88.60	89.62	89.59

remaining available. In contrast, server No. 4 and No. 9 receive a light workload with nearly 19 percentage of total capacity remained unused. This leads to the unfavourable situation where some heavy load users like No.5 and No.8 delay the response time while abundant resources still remain wasted on some light load serves like No. 4 and No. 9.

As a comparison, users reassign their jobs with their new strategies obtained from our 1-D mean field evolutionary approach. Consequently, light load servers like No. 4 and No. 9 obtain more jobs from users reducing the burden on heavy load users like No. 5 and No. 8. However, the workload on servers are not exactly the same because servers have different total serving capacity.

3) Comparison With Other Algorithms: In this subsection, we compare our mean field evolutionary approach (MFEA) with two well-known job scheduling algorithms. They are proportional scheme (PROP) [27] and global optimization scheme (GOS) [24]. PROP will assign users' jobs proportionally to the servers based on their service rate. It has a low computational cost but not the optimal result. Therefore, PROP can be regarded as a suitable baseline. GOS tries to allocate a disproportionately higher fraction of jobs to the more powerful servers. It has higher computational cost but it can obtain the optimal strategy to minimize the overall response time. In Fig. 5a, we see that our approach converges faster than both GOS and PROP and three methods achieve similar overall response time with different job scheduling strategy when $\theta \approx 90\%$. In Fig. 5b, our approach also converge faster than GOS and PROP. However, Gos and PROP achieve better fairness of the system as shown in the zoom plot of Fig. 5b.

In Table II, we compare the utilization rate of 10 servers with $\theta \approx 90\%$. As we mentioned above, GOS assign heavier workload to the more powerful servers, like server No. 4 and server No. 9. The utilization rate of servers are exactly the same as the utilization rate of whole system in PROP. The utilization rate distribution of our approach is very close to that of GOS as shown in Table II.

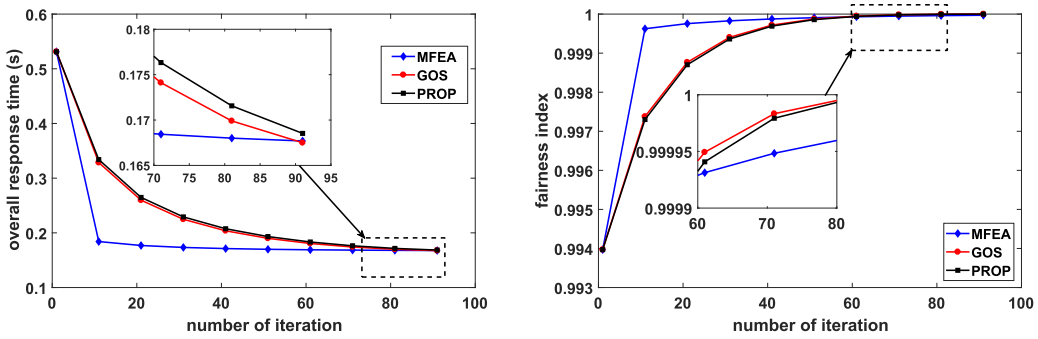
C. Performance Analysis for the Multi-Resource Case

In this subsection, we show the efficiency of our 2-D mean field evolutionary approach in the multi-resource case. The results are similar to single-resource case so our analysis becomes more concise and we do not compare our algorithm with other algorithms here, because, to the best of our knowledge, a multi-resource case in a dense-user network has not been considered by previous literature.

1) Convergence Analysis: In Fig. 6, similar convergence patterns of three performance indicators has been shown. The overall response time converges much slower compared with the single-resource case because the gradient flow on 2-D strategy graph, which has been shown in Fig. 2, increases the computational cost and more iterations are needed to find the optimal strategies for users.

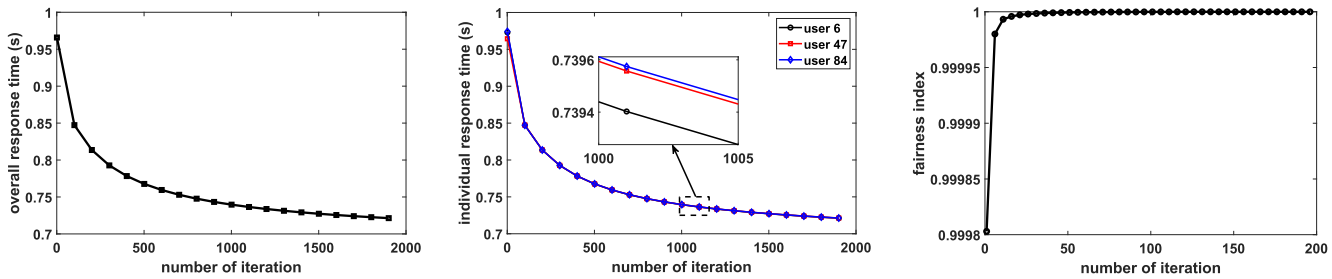
2) Load Balance Analysis: In Table III, we choose three representative servers and four representative resources to show our approach's efficiency of load balance in the multi-resource case. Right after initialization, utilization rate of resource A on server No. 1 is high while much idle

Authorized licensed use limited to: University of Houston. Downloaded on February 08,2021 at 01:41:05 UTC from IEEE Xplore. Restrictions apply.



(a) Comparison on overall response time (b) Comparison on fairness index

Fig. 5. Comparison with GOS and PROP.



(a) Overall response time (b) Individual response time (c) Fairness index

Fig. 6. Convergence analysis of multi-resource case.

TABLE III
UTILIZATION RATE OF DIFFERENT RESOURCES

Server number	Utilization rate of resources on servers (%) (before/after)			
	A	B	C	D
1	94.23/90.02	89.88/90.32	90.06/89.69	85.05/90.35
4	85.16/90.05	94.35/89.14	91.15/88.77	91.69/89.77
7	93.70/89.30	90.31/88.05	85.25/89.70	87.48/89.71

resource A is wasted on server No. 4. Besides, utilization rate of resource B on server No. 1 is low while server No. 4 is crowded with jobs needed resource B. That is why the overall MEC system has a bad performance at the beginning. When mobile users find their new strategies to assign their jobs to different servers, the workload of different resources on different servers gradually balanced. They are not exactly the same because we are considering heterogeneous servers (the service rates are different) and the difference between them is small because we are considering a rather limited resource case.

D. Scalability Analysis

As shown in Algorithm 1, during each iteration, we first construct the gradient matrix Λ_i for user i based on the (21), (22) and (23). For each server (node) in the middle of the 1-D strategy graph in Fig. 2, 3 steps are needed to calculate the relevant gradient. For the first and the last server (node), 2 steps are needed to calculate the relevant gradient. Therefore, the total steps to compute the gradient matrix Λ_i is $3s-2$. Then we need to calculate the gradient matrix for all u users, which means the time complexity for Algorithm 1 is $\mathcal{O}(su)$.

As shown in Algorithm 2, during each iteration, we also construct a gradient matrix for Λ_i for each user i based on Theorem 1. As shown in the 2-D strategy graph in Fig. 2, the nodes need to be divided into three types (vertex nodes, edge nodes, and interior nodes) according to the number of their neighbors. The total steps for the gradient matrix construction is $4rs-16$. Then we repeat this for all u users, which means the time complexity for Algorithm 2 is $\mathcal{O}(rsu)$.

From the above analysis, the time complexity for our mean field evolutionary approach is $\mathcal{O}(rsu)$, where r is the number of resources available on each server, s is the number of servers, and u is the number of users.

VI. CONCLUSION

In this paper, we have modeled the load balancing problems for both the single-resource case and multi-resource case in a dense-user MEC system. Interactions between a large number of users will result in computational complexity for traditional games because they usually deal with each user separately. However, a mean field game will regard these users as a "mean field" when analyzing the behavior of one generic user. Therefore, we have formulated these problems as non-cooperative population games and solved them with a innovative mean field evolutionary approach. Moreover, to handle the problems under these two cases, we have designed two different strategy graphs for our approach. Furthermore, we have conducted a comprehensive simulation and evaluated the performance of

Authorized licensed use limited to: University of Houston. Downloaded on February 08,2021 at 01:41:05 UTC from IEEE Xplore. Restrictions apply.

our algorithm in terms of three performance indicators: overall response time, individual response time, and fairness index. The simulation results show that our approach converges faster than existing methods and achieves performance very close to the global optimal solution in the single-resource case. Moreover, our approach also improves the performance of the system significantly in the multi-resource case.

REFERENCES

[1] H. Gao, W. Li, R. A. Banez, Z. Han, and H. V. Poor, "Mean field evolutionary dynamics in ultra dense mobile edge computing systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[3] (Feb. 2019). *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual- networking-index-vni/white-paper-c11-738429.html

[4] Z. Han, D. Niyato, W. Saad, and T. Başar, *Game Theory for Next Generation Wireless and Communication Networks: Modeling, Analysis, and Design*. Cambridge, U.K.: Cambridge Univ. Press, 2019.

[5] D. T. Hoang, D. Niyato, D. N. Nguyen, E. Dutkiewicz, P. Wang, and Z. Han, "A dynamic edge caching framework for mobile 5G networks," *IEEE Wireless Commun.*, vol. 25, no. 5, pp. 95–103, Oct. 2018.

[6] Y. Zhang, C. Yang, J. Li, and Z. Han, "Distributed interference-aware traffic offloading and power control in ultra-dense networks: Mean field game with dominating player," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8814–8826, Sep. 2019.

[7] Y. Teng, M. Liu, F. R. Yu, V. C. M. Leung, M. Song, and Y. Zhang, "Resource allocation for ultra-dense networks: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2134–2168, 3rd Quart., 2019.

[8] A. Kishor and R. Niyogi, "A population-based approach for load balancing in distributed computing systems," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Bangalore, India, Sep. 2018, pp. 311–317.

[9] D. Mosk-Aoyama, T. Roughgarden, and D. Shah, "Fully distributed algorithms for convex optimization problems," *SIAM J. Optim.*, vol. 20, no. 6, pp. 3260–3279, Jan. 2010.

[10] D. Grosu and A. T. Chronopoulos, "Noncooperative load balancing in distributed systems," *J. Parallel Distrib. Comput.*, vol. 65, no. 9, pp. 1022–1034, Sep. 2005.

[11] R. Tripathi, S. Vignesh, V. Tamarapalli, A. T. Chronopoulos, and H. Siar, "Non-cooperative power and latency aware load balancing in distributed data centers," *J. Parallel Distrib. Comput.*, vol. 107, pp. 76–86, Sep. 2017.

[12] T. Roughgarden, "Stackelberg scheduling strategies," *SIAM J. Comput.*, vol. 33, no. 2, pp. 332–350, Jan. 2004.

[13] B. Yang, Z. Li, S. Chen, T. Wang, and K. Li, "Stackelberg game approach for energy-aware resource allocation in data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3646–3658, Dec. 2016.

[14] C. Yang, J. Li, M. Sheng, A. Anpalagan, and J. Xiao, "Mean field game-theoretic framework for interference and energy-aware control in 5G ultra-dense networks," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 114–121, Feb. 2018.

[15] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[16] R. A. Banez, L. Li, C. Yang, L. Song, and Z. Han, "A mean-field-type game approach to computation offloading in mobile edge computing networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019, pp. 1–6.

[17] T. Q. Duong, X. Chu, and H. A. Suraweera, *Mean Field Games for 5G Ultra-Dense Networks: A Resource Management Perspective*. Hoboken, NJ, USA: Wiley, 2019, pp. 65–89.

[18] T. D. Braun, H. J. Siegel, A. A. Maciejewski, and Y. Hong, "Static resource allocation for heterogeneous computing environments with tasks having dependencies, priorities, deadlines, and multiple versions," *J. Parallel Distrib. Comput.*, vol. 68, no. 11, pp. 1504–1516, 2008.

[19] R. A. Banez, H. Gao, L. Li, C. Yang, Z. Han, and H. V. Poor, "Belief and opinion evolution in social networks based on a multi-population mean field game approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–6.

[20] A. Bensoussan, T. Huang, and M. Laurière, "Mean field control and mean field game models with several populations," 2018, *arXiv:1810.00783*. [Online]. Available: http://arxiv.org/abs/1810.00783

[21] K. S. Trivedi, *Probability and Statistics With Reliability, Queuing, and Computer Science Applications*. Hoboken, NJ, USA: Wiley, 2016.

[22] S. Penmatsa and A. T. Chronopoulos, "Game-theoretic static load balancing for distributed systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 4, pp. 537–555, Apr. 2011.

[23] H. Kameda, *Optimal Load Balancing in Distributed Computer Systems*. Cham, Switzerland: Springer, 1997.

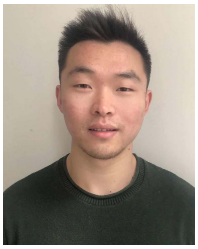
[24] X. Tang and S. T. Chanson, "Optimizing static job scheduling in a network of heterogeneous computers," in *Proc. Int. Conf. Parallel Process.*, Toronto, ON, Canada, 2000, pp. 373–382.

[25] S.-N. Chow, W. Li, J. Lu, and H. Zhou, "Equilibrium selection via optimal transport," 2017, *arXiv:1703.08442*. [Online]. Available: http://arxiv.org/abs/1703.08442

[26] S.-N. Chow, W. Li, J. Lu, and H. Zhou, "Population games and discrete optimal transport," *J. Nonlinear Sci.*, vol. 29, no. 3, pp. 871–896, Oct. 2018.

[27] Y.-C. Chow and Kohle, "Models for dynamic load balancing in a heterogeneous multiple processor system," *IEEE Trans. Comput.*, vol. C-28, no. 5, pp. 354–361, May 1979.

[28] N. Raveendran, Y. Zha, Y. Zhang, X. Liu, and Z. Han, "Virtual core network resource allocation in 5G systems using three-sided matching," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019, pp. 1–6.



Hao Gao (Graduate Student Member, IEEE) received the B.E. degree in electrical and information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2018. He is currently pursuing the Ph.D. degree in electrical engineering with the University of Houston, USA.

His current research interests include mean field game and related applications in wireless communication.

Wuchen Li received the B.Sc. degree in mathematics from Shandong University in 2009, the M.S. degree in statistics, and the Ph.D. degree in mathematics from the Georgia Institute of Technology in 2016.

He was a CAM Assistant Adjunct Professor with the Department of Mathematics, University of California at Los Angeles, Los Angeles, from 2016 to 2020. He is currently an Assistant Professor with the University of South Carolina. His research interests include optimal transport, information geometry, and mean field games with applications in data science.

Reginald A. Banez received the B.S. degree in electronics and communications engineering from Maṕua University, Manila, Philippines, in 2008, and the M.S. and Ph.D. degrees in electrical engineering from the University of Houston, Houston, TX, USA, in 2015 and 2020, respectively.

His research interests include communication networks, mean field games, and machine learning.

Authorized licensed use limited to: University of Houston. Downloaded on February 08,2021 at 01:41:05 UTC from IEEE Xplore. Restrictions apply.

Zhu Han (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively.

From 2000 to 2002, he was a Research and Development Engineer at JDSU, Germantown, MD, USA. From 2003 to 2006, he was a Research Associate with the University of Maryland. From 2006 to 2008, he was an Assistant Professor with Boise State University, ID, USA. He is currently a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department and with the Computer Science Department, University of Houston, TX, USA. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. He received the NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the *Journal on Advances in Signal Processing* in 2015, the IEEE Leonard G. Abraham Prize in the field of Communications Systems (Best Paper Award in the IEEE JSAC) in 2016, and several best paper awards in IEEE conferences. He is also a recipient of the 2021 IEEE Kiyo Tomiyasu Award, for outstanding early to mid-career contributions to technologies holding the promise of innovative applications, with the following citation: "for contributions to game theory and distributed management of autonomous communication networks." He was an IEEE Communications Society Distinguished Lecturer from 2015 to 2018, has been an AAAS Fellow since 2019, and has been an ACM Distinguished Member since 2019. He has been a 1% Highly Cited Researcher since 2017 according to Web of Science.

H. Vincent Poor (Life Fellow, IEEE) received the Ph.D. degree in EECS from Princeton University in 1977.

From 1977 to 1990, he was on the faculty of the University of Illinois at Urbana–Champaign. Since 1990, he has been on the faculty at Princeton University, where he is currently the Michael Henry Strater University Professor of electrical engineering. From 2006 to 2016, he served as the Dean of the Princeton's School of Engineering and Applied Science. He has also held visiting appointments at several other universities, including most recently at Berkeley and Cambridge. His research interests include information theory, signal processing, and machine learning and their applications in wireless networks, energy systems, and related fields. Among his publications in these areas is the recent book *Multiple Access Techniques for 5G Wireless Networks and Beyond* (Springer, 2019). He is a member of the National Academy of Engineering and the National Academy of Sciences and is a Foreign Member of the Chinese Academy of Sciences, the Royal Society, and other national and international academies. Recent recognition of his work includes the 2017 IEEE Alexander Graham Bell Medal and a D.Eng. honoris causa from the University of Waterloo awarded in 2019.

Authorized licensed use limited to: University of Houston. Downloaded on February 08,2021 at 01:41:05 UTC from IEEE Xplore. Restrictions apply.