Socially Aware Dynamic Computation Offloading Scheme for Fog Computing System With Energy Harvesting Devices

Liqing Liu, Zheng Chang, *Senior Member, IEEE*, and Xijuan Guo

***Abstract*—Fog computing is considered as a promising technology to meet the ever-increasing computation requests from a wide variety of mobile applications. By offloading the computation-intensive requests to the fog node or the central cloud, the performance of the applications, such as energy consumption and delay, are able to be significantly enhanced. Meanwhile, utilizing the recent advances of social network and energy harvesting (EH) techniques, the system performance could be further improved. In this paper, we take the social relationships of the EH mobile devices (MDs) into the design of computational offloading scheme in fog computing. With the objective to minimize the social group execution cost, we advocate game theoretic approach and propose a dynamic computation offloading scheme designing the offloading process in fog computing system with EH MDs. Different queue models are applied to model the energy cost and delay performance. It can be seen that the proposed problem can be formulated as a generalized Nash equilibrium problem (GNEP) and we can use exponential penalty function method to transform the original GNEP into a classical Nash equilibrium problem and address it with semi-smooth Newton method with Armijo line search. The simulation results demonstrate the effectiveness of the proposed scheme.**

***Index Terms*—Computation offloading, energy consumption, energy harvesting (EH), execution cost, fog computing, generalized Nash equilibrium problem (GNEP), social-aware mobile network.**

I. Introduction

A. Background and Motivation

MOBILE device (MD) has become an indispensable part of our daily life as they can provide convenient communication almost anytime and anywhere. The mobile application markets are also triggered by the advanced mobile technologies and high data rate wireless networks. However, due to the resource and battery life restrictions, the gap between the limited computing capability and demand for executing complex applications is gradually increasing. Many computational-intensive and latency-sensitive mobile applications have poor performance when they are performed on smart phones, such as image processing, chess gaming, etc [1].

Recent study shows that mobile cloud computing (MCC) technology provides a promising opportunity to overcome the limitation of hardware and save energy for MDs by offloading the computational-intensive tasks to the cloud for execution [2]. To date, several types of mobile cloud architectures are categorized [3], such as the traditional central cloud, ad hoc mobile cloud, cloudlet, etc. The traditional central cloud can provide huge storage, high computation power, as well as reliable security. However, it is worth mentioning that the traditional central cloud is usually remotely located and far away from their users, thus long latency may be incurred. Therefore, in some cases, the distant cloud may not be desirable for latency-sensitive mobile applications [2]. To overcome these problems, fog computing, also known as "cloud at the edge," emerges as an alternative proximity solution to provide pervasive and agile computation services for the MDs and support future cloud services and applications, especially to the Internet-of-Things applications with strict requirement of latency and high resilience [4]. Fog computing can provide computing resources at the edge of radio access networks. The idea of using fog computing brings both computational and radio resource more closer to the MDs, thus improving scalability in both computation and radio aspects [5], [6]. However, it can be noticed that the computational resource in the fog cannot be treated as sufficiently as the traditional central cloud, as it is usually targeted to serve a smaller portion of users [6].

Although computation offloading is an effective solution for resource-limited MDs to use the powerful computational resources at cloud servers. But for conventional battery-powered MDs, the computation performance may be discounted due to lack of sufficient battery energy, i.e., the being executed applications have to be terminated when the battery energy is exhausted. Certainly, this can possibly be solved by increasing the battery capacity or recharging the batteries from time to time. However, due to the restrictions of the MDs on size, weight, location, ergonomics, heat dissipation, and so on, frequent recharging is hard to be realized in many cases. Energy harvesting (EH) is seen as a promising and green technology to resolve these issues, which can enables the MDs to harvest energy from multiple aspects [7], [8], such as from environmental energy sources such as wind energy and solar, from human motion, and from ambient radio signals, e.g., RF

Manuscript received November 20, 2017; revised February 18, 2018; accepted March 9, 2018. This work was supported in part by the Academy of Finland under Decision 284748 and in part by the NSF of Hebei under Grant E2017203351. *(Corresponding author: Zheng Chang.)*

L. Liu and X. Guo are with the College of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China (e-mail: liuliqing_yanyan@163.com; xjguo@ysu.edu.cn).

Z. Chang is with the Faculty of Information Technology, University of Jyväskylä, 40014 Jyväskylä, Finland (e-mail: zheng.chang@jyu.fi).

EH. The harvested energy can be used for locally operation, task execution, and request offloading. With EH, the battery time of the MDs can be extended, and self-sustaining can be achieved [9]–[11].

It can be observed that most of related work mainly focus on the case where each mobile user is self-interested and aims to minimize its own cost [2], [12], [13]. However, recent investigations on social networks show that there is a strong connection between the user social relations and its behavior [14]–[19]. Due to the existence of social relationships, it is natural that the mobile user not only cares about his own utility, but also pay attention to the utility of his neighbors with social relations, such as families, friends, etc. Such a observation motivates us to exploit the social ties among mobile users to achieve mutually beneficial computation offloading decision-making and correspondingly, improve the system-level performance.

B. Contribution

In this paper, we consider the MDs are with EH capabilities and are interested in offloading the computation task to the fog node. Considering the aforementioned aspects, in this paper, we aim to take the social relationships of the mobile users into the design of computation offloading scheme. With the objective to minimize the social group execution cost, we advocate game theoretic approach and propose a dynamic computation offloading scheme. More specifically, our major contributions are summarized as follows.

1) In the considered system, with the objective to minimize the social group execution cost, we propose a socially aware dynamic computation offloading algorithm with the consideration of EH devices. Game theory is employed to model the interactions within the group.
2) In particular, different queue models are applied to derive the delay performance during the offloading process in our model. The requests generated from each MD are assumed to follow Poisson process. The requests process queue at the MD is considered as a $M/M/1$ queue, the one in the fog is considered as a $M/G/1$ queue, and the one at the central cloud is considered as a $M/G/\infty$ queue, which is consistent with common sense and sparely studied in the previous work about MCC.
3) The algorithm to solve the proposed generalized Nash equilibrium problem (GNEP) is novel. First, we use exponential penalty function method to penalize the coupling constraints and transform the original GNEP into a classical NEP. Second, we formulate the Karush–Kuhn–Tucker (KKT) conditions for the smoothing penalized NEPs into a system of nonsmooth equations, and then apply the semi-smooth Newton method with Armijo line search to solve the system. The proposed one can reduce the accumulated error and improve the calculation accuracy during iteration process effectively.
4) Extensive simulations are conducted to evaluate the effectiveness of the presented scheme. It is shown that our scheme can find the optimal decision strategy at a certain request arrival rate at each time slot for each MD, and obtain the minimum value of average social group execution cost.

C. Organization

The reminder of this paper is organized as follows. We first briefly overview the related works in Section II. In Section III, the system model is introduced and the problem is formulated in Section IV. In Section V, we propose a semi-smooth Newton method to solve the formulated problem. The simulation results are presented and discussed in Section VI, and finally, we conclude this paper in Section VII.

II. Related Work

Most of the papers in MCC focus on designing different and effective offloading schemes for the resource-limited MDs to offload the computation to the central cloud. Zhang *et al.* [12] investigated collaborative task execution problem for mobile applications. The authors aim to minimize the energy consumption on the MD while meeting a time deadline, by strategically offloading tasks to the cloud. Zhang *et al.* [13] proposed a theoretical framework of energy-optimal MCC under stochastic wireless channel. Deng *et al.* [2] considered a mobile computation offloading problem where multiple mobile services in workflows can be invoked to fulfill their complex requirements and the decisions can be made on whether the services of a workflow should be offloaded.

Meanwhile, fog computing is a new concept emerged in recent years and provides pervasive and agile computation augmenting services for the MDs with short delay [4]–[6]. Deng *et al.* [5] studied the multiuser computation offloading problem for fog computing in a multichannel wireless interference environment. Sardellitti *et al.* [6] considered an MIMO multicell fog computing system where multiple mobile users ask for computation offloading to a fog cloud server. The authors formulate the offloading problem as the joint optimization of the radio resources and computational resources to minimize the overall users' energy consumption, while meeting latency constraints.

EH is integrated with communication systems to enable its ability of obtaining energy efficient communications and self-sustainable [7]–[11]. Realizing the EH in cloud computing and cellular network is an emerging technique to enhance the sustain-ability of battery-powered network elements and has attracted great attention from the research community. For example, Mao *et al.* [9] investigated a green mobile edge cloud computing system with EH devices and develop a Lyapunov optimization-based dynamic computation offloading algorithm, which jointly decides the offloading decision, the CPU-cycle frequencies for mobile execution, and the transmit power for computation offloading. Yang *et al.* [11] proposed a D2D communication heterogeneous cellular network based on EH, where mobile user equipments harvest energy from the base station (BS) or the access point and use the harvested energy for D2D communication. Meanwhile, social networks and their applications have been explored in the design of wireless networks. The concept of the user social pattern has been

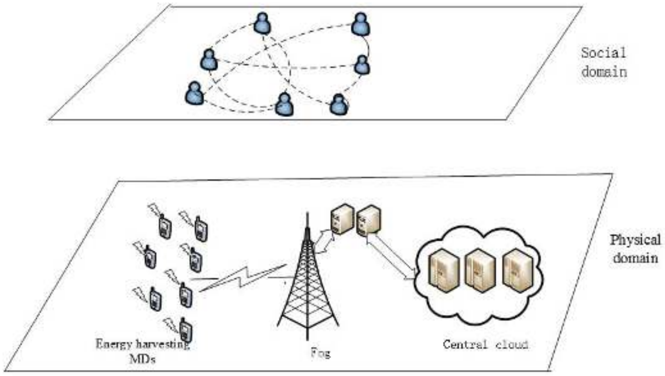

Fig. 1. System model.

characterized in long term evolution-advanced heterogeneous networks to enhance the energy efficiency and network spectrum. For mobile networking, exploiting mobile users' social relationship turn out to be a new field for network optimization and design [16], [17]. For example, Zhang *et al.* [16] studied a social-aware approach for optimizing throughput of D2D communication by exploiting the information from both social network layer and the physical wireless network layer.

III. System Model

As shown in Fig. 1, the system consists of N MDs with social relations, fog node, and a control cloud. All the MDs are equipped with EH capabilities. The harvested energy can be stored in the battery and used for local execution or data transmission. Each MD executes an application and generates a series of homogeneous service requests which are independent of each other. Each MD contains one processor, a single server first-in-first-out (FIFO) queue to store arriving requests pending for execution, and the wireless interfaces to connect wireless network. The fog node is deployed to release the load on central cloud and bring low service latency to the nearby users. In this paper, we apply queueing theory to model the service latency. Queuing theory has been widely used in the analysis of resource contention in the communication and computing systems, and it is a natural candidate to capture the main features of our systems. In our system, the process queue at the MD is considered as a $M/M/1$ queue, which amounts to assimilate the task arrival process to a Poisson process. The fog node comprises multiple servers, and with a dispatcher that can uniformly distribute the arrival data among the servers. Correspondingly, the queue model of fog node is considered as a $M/G/1$ queue [21]. The central cloud hosts infinite servers in the remote data centers, with no contention among different users, and we model the process as a $M/G/\infty$ queue. We assume that the time is slotted and the length of each time slot is τ. We also denote the time slot and the time slot index set by t and $\mathcal{T} = \{0, 1, \dots, t, \dots, T-1\}$.

We assume that the requests generated from MD $i, i \in \mathcal{N}, \mathcal{N} = \{1, 2, \dots, N\}$ follow a Poisson process with an average arrival rate of $\lambda_{i,t}$ at each time slot t [20]. The requests are assume to be delay sensitive, mutually independent. Each request generating from the MD i contains a data size of θ_i. Each computation request can be executed locally at the MD, or be offloaded to the fog or the central cloud. In addition, none of the above three computation modes may happen, e.g., when the MD does not have sufficient energy, some generated computation requests have to be dropped. The generated requests can be allocated to the local processor, the fog and the central cloud, or even dropped out in parallel at the beginning of the next time slot. So throughout this paper, "at time slot t" means the requests are generated at time slot t but executed at time slot $t+1$. The decision of MD i at time slot t is modeled as a triple $p_{i,t} = (p^M_{i,t}, p^F_{i,t}, p^C_{i,t}, p^D_{i,t})$, where $p^M_{i,t} + p^F_{i,t} + p^C_{i,t} + p^D_{i,t} = 1$, which represents the proportion that the requests are executed locally ($p^M_{i,t}$), offloaded to the fog ($p^F_{i,t}$), offloaded to the central cloud ($p^C_{i,t}$), or be dropped ($p^D_{i,t}$) at each time slot t. The MDs compete for the computing resource in the fog in order to minimize their execution cost. They are allowed to make their own strategies without a central authority but can obtain information from a cloud controller. Game theory is employed to model this interaction, where the MDs play with the obtained information from the cloud controller, until they reach a stable state, i.e., a Nash equilibrium. For ease of reference, we list all the key notations used in our system model in Table I.

TABLE I
Summary of the Key Notations

Notations	Meanings
$\mathcal{N}$	the set of energy harvesting MDs
$\mathcal{T}$	the set of time slots
$p^M_{i,t}$	the percentage that MD i executed locally at time slot t
$p^F_{i,t}$	the percentage that MD i offloaded to fog at time slot t
$p^C_{i,t}$	the percentage that MD i offloaded to cloud at time slot t
$p^D_{i,t}$	the percentage that MD i dropped at time slot t
$\boldsymbol{p_{i,t}}$	the decision strategy of MD i at time slot t
$\boldsymbol{p_{\bar{i},t}}$	the strategies vector other than i at time slot t
$\boldsymbol{p_i}$	strategies vector of MD i of all the time slots
$\boldsymbol{p_{\bar{i}}}$	strategies vector other than MD i of all the time slots
$\boldsymbol{p}$	strategies vector of all MDs of all the time slots
$\lambda_{i,t}$	the request rate of MD i at time slot t
u^M_i	the computing capability of MD i
$l^M_{i,t}$	the workload of MD i at time slot t
κ_i	the per cycle energy consumption of MD i
θ_i	the data size in each request of MD i
x_i	the CPU cycles required for per bit data of MD i
μ_i	the per rate punishment cost of MD i
W	the channel bandwidth
$q_{i,t}$	the transmission power of MD i at time slot t
$g^{BS}_{i,t}$	the channel gain for MD i at time slot t
$\omega_{i,t}$	the noise power at time slot t
c	the number of servers in the fog
u^F	the server service rate in the fog
l^F_t	the workload of the fog at time slot t
T^{FC}	the fixed delay from fog to the central cloud
u^C	the service rate of the central cloud
$e_{i,t}$	the harvested energy for MD i at time slot t
$B_{i,t}$	the total energy for MD i at time slot t
μ_i	the punishment for per dropped task for MD i
$\bar{\alpha}$	the weight of task dropping punishment

A. Local Execution Model

Let u^M_i denotes the computing capability of MD i, which is determined by the intrinsic nature of the MD, i.e., CPU cycle. Different MDs may have different computing capability. Additionally, we assume that $l_{i,t}^{M}$ denotes the normalized workload on the MD i at time slot t, which represents the percentages of CPU that have been occupied. $l_{i,t}^{M} = 0$ indicates that the CPU is totally idle at time slot t. When considering a $M/M/1$ queue, the response time of a $M/M/1$ queue is $R = [(1/u)/(1 - \rho)]$ [22], where $\rho = (\lambda/u)$ is the queue utilization, λ is the arrival rate, u is the service rate. Accordingly, the average response time $T_{i,t}^{M}(p_{i,t}^{M})$ for locally processing requests at MD i is expressed as follows:

$$T_{i,t}^{M}\left(p_{i,t}^{M}\right) = \frac{1/u_i^M\left(1 - l_{i,t}^{M}\right)}{1 - \frac{\lambda_{i,t}p_{i,t}^{M}}{u_i^M\left(1-l_{i,t}^{M}\right)}} = \frac{1}{u_i^M\left(1 - l_{i,t}^{M}\right) - \lambda_{i,t}p_{i,t}^{M}}. \quad (1)$$

Assume that the number of CPU cycles needed for computing 1-bit of input data locally is x_i and the energy consumption per cycle is κ_i for MD i. Then the expression $x_i\kappa_i$ is the per bit computing energy consumption for MD i. x_i and κ_i are related to the intrinsic nature of the CPU and the complexity of the requests for each MD. Then the energy consumption $E_{i,t}^{M}(p_{i,t}^{M})$ of local executing the requests for MD i can be given as follows:

$$E_{i,t}^{M}\left(p_{i,t}^{M}\right) = x_i\kappa_i p_{i,t}^{M}\lambda_{i,t}\tau\theta_i. \quad (2)$$

B. Fog Execution Model

MD i transmits the data to the fog through a BS at the beginning of the time slot with rational considerations. The wireless channel is assumed to be independent and identically distributed (i.i.d) block fading, i.e., the channel remains static within each time slot, but varies from one to another. Considering the mutual interference caused by other MDs in the system and the nonignorable background interference, we can obtain the uplink transmission rate for computation offloading of MD i at time slot t as follows [5]:

$$R_{i,t} = W\log_2\left(1 + \frac{q_{i,t}g_{i,t}^{BS}}{w_{i,t} + \sum_{j\in N, j\neq i} q_{j,t}g_{j,t}^{BS}}\right) \quad (3)$$

where W is the channel bandwidth, $q_{i,t}$ is the transmission power of the MD i at time slot t, $g_{i,t}^{BS}$ is the channel gain between the MD i and BS at time slot t, $\omega_{i,t}$ is the background noise interference received by MD i at time slot t.

From (3), we can then obtain the uplink transmission time $T_{i,t}^{\mathrm{UP}}$ of MD i for offloading the data to BS as follows:

$$T_{i,t}^{\mathrm{UP}}\left(p_{i,t}^{F}, p_{i,t}^{C}\right) = \frac{\left(p_{i,t}^{F} + p_{i,t}^{C}\right)\lambda_{i,t}\tau\theta_i}{R_{i,t}} = \frac{\left(p_{i,t}^{F} + p_{i,t}^{C}\right)\lambda_{i,t}\tau\theta_i}{W\log_2\left(1 + \frac{q_{i,t}g_{i,t}^{BS}}{w_{i,t}+\sum_{j\in N, j\neq i} q_{j,t}g_{j,t}^{BS}}\right)}. \quad (4)$$

Then, the energy consumption of uplink transmission $E_{i,t}^{\mathrm{UP}}(p_{i,t}^{F}, p_{i,t}^{C})$ can be given as follows:

$$E_{i,t}^{\mathrm{UP}}\left(p_{i,t}^{F}, p_{i,t}^{C}\right) = q_{i,t}T_{i,t}^{\mathrm{UP}}\left(p_{i,t}^{F}, p_{i,t}^{C}\right) = \frac{q_{i,t}\left(p_{i,t}^{F} + p_{i,t}^{C}\right)\lambda_{i,t}\tau\theta_i}{W\log_2\left(1 + \frac{q_{i,t}g_{i,t}^{BS}}{w_{i,t}+\sum_{j\in N, j\neq i} q_{j,t}g_{j,t}^{BS}}\right)}. \quad (5)$$

The fog node is located at the BS and connects to the BS through fiber with large enough bandwidth, so we just neglect the transmission time from the BS to fog node. Accordingly, we assume that there are c homogeneous servers deployed in the fog, and the service rate of each server is denoted as u^F. The requests from different MDs in the system are pooled together with a total rate $\lambda_{\mathrm{total},t}$ at time slot t which also follows Poisson process. Therefore, $\lambda_{\mathrm{total},t}$ is given as follows:

$$\lambda_{\mathrm{total},t} = \sum_{i=1}^{N} p_{i,t}^{F}\lambda_{i,t}. \quad (6)$$

Correspondingly, we assume that the workload of the fog node is denoted as $l_t^F (0 < l_t^F < 1)$, which is the average occupied percentage of each server at time slot t. As a $M/G/1$ queue is considered at the fog node, the average response time at the fog as follows [21], [23]:

$$T_{F,t}\left(p_{i,t}^{F}\right) = \frac{2u^F\left(1 - l_t^F\right) - \sum_{i=1}^{N}\lambda_{i,t}p_{i,t}^{F}/c}{2u^F\left(1 - l_t^F\right)\left[u^F\left(1 - l_t^F\right) - \sum_{i=1}^{N}\lambda_{i,t}p_{i,t}^{F}/c\right]}. \quad (7)$$

After the execution is done at the fog node, the results will be delivered to the MDs. And we neglect the time and energy consumption for the MDs to receive the processed requests outcome, due to the fact that for many applications, for example, the face recognition, the size of the computation output in general is much smaller than the size of computation input data [5], [7].

C. Cloud Execution Model

Additionally, we assume that there is a fixed delay T_{FC} for sending the requests to the central cloud through the fog. As the central cloud has sufficient computing resources to process these requests, the queuing time of the requests in the central cloud can be negligible. The queue model at the central cloud is considered as $M/G/\infty$ with the service rate u^C, which is usually faster than the fog service rate u^F. Then, the response time $T_{C,t}(p_{i,t}^{C})$ of the requests offloaded to the central cloud can be presented as follows:

$$T_{C,t}\left(p_{i,t}^{C}\right) = \frac{1}{u^C}. \quad (8)$$

When the execution at central cloud is done, the results will be delivered to fog and then delivered to the MDs. Similarly, the time and energy consumption for receiving the results for MDs can be neglected.

D. EH Model

A successive energy packet arrival model is used to model the EH process. We assume that the arrival of energy packet also follows a Poisson process with an average arrival rate $e_{i,t}$, and $e_{i,t} \le e_{i,t}^{\mathrm{th}}$, we assume that $e_{i,t}^{\mathrm{th}}$ is the maximum energy arrival rate, and it is i.i.d in different time slots. The arrived energy will be harvested and stored in the battery, and used for either local execution or computation offloading. Let $B_{i,t}$ denotes the battery energy level for MD i at the beginning of time slot t. Without loss of generality, we assume $B_{i,t} < \infty$, $\forall t \in T$. In this paper, we just ignore the energy consumption for other purposes besides local computation and data transmission. Denote the energy consumed by the MD i in time slot t as $E_{i,t}$, which comprises of two parts: 1) energy consumption of local service request processing and 2) energy consumption for sending requests, which depends on the strategy it chooses. We can express $E_{i,t}$ as follows:

$$\begin{aligned} E_{i,t} &= E_{i,t}^{M}\left(p_{i,t}^{M}\right) + E_{i,t}^{\mathrm{UP}}\left(p_{i,t}^{F}, p_{i,t}^{C}\right) \\ &= x_i \kappa_i p_{i,t}^{M} \lambda_{i,t} \tau \theta_i + \frac{q_{i,t}\left(p_{i,t}^{F} + p_{i,t}^{C}\right)\lambda_{i,t}\tau\theta_i}{W\log_2\left(1 + \frac{q_{i,t}g_{i,t}^{BS}}{w_{i,t} + \sum_{j\in N, j\neq i} q_{j,t}g_{j,t}^{BS}}\right)}. \end{aligned} \tag{9}$$

$E_{i,t}$ should be smaller than the battery level, i.e.,

$$E_{i,t} \le B_{i,t} \quad \forall t \in T. \tag{10}$$

Thus, the battery energy level of MD i evolves according to the following equation:

$$B_{i,t+1} = B_{i,t} - E_{i,t} + e_{i,t} \quad \forall t \in T. \tag{11}$$

It is much more complicated to design the computation offloading policies for the fog computing system with the MDs equipped with EH function, compared to the conventional MCC systems with battery-powered MDs. Moreover, the system decisions are coupled among different time slots because of the temporally evolved battery energy level. Consequently, it is very challenge to determine the optimal computation offloading strategies, which would balance the computation performances of the current and future computation requests as better as possible.

E. Social Network Model

In this section, we use a social graph $(\mathcal{N}, \varepsilon^s)$ to denote the social tie structure among the MDs. The vertex set is the $\mathcal{N}$ MD members and the edge set is denoted as $\varepsilon^s = \{(i,j) : e_{i,j}^{s} = 1, \forall i, j \in N\}$, where $e_{i,j}^{s} = 1$ denotes that MD i and j have social relationship between each other, and verse vice. The strength of social relationship between MD i and MD j is denoted as s_{ij}, which is normalized to be $s_{ij} \in [0, 1]$. And the larger the value of s_{ij}, the stronger the social tie between the two MDs. The MD i's social tie to itself is $s_{ii} = 1$. N_i^S is defined as the MD i's social group, which is the set of MDs that have social ties with MD i, i.e., $N_i^S \triangleq \{j \in N | e_{ij}^{s} \in \varepsilon^s\}$.

It is worth noting that the social relationships among MDs can be obtained by locally putting forward the recognition process through the proximity communications technology, for example, the WiFi-direct and so on, prior to the computation offloading process.

IV. Problem Formulation

In this section, the execution cost is defined as the weighted sum of the execution delay and the task dropping punishment cost, which will be described specifically as follows. The average execution delay for MD i at time slot t can be denoted as follows:

$$\begin{aligned} &T_{i,t}\left(\boldsymbol{p}_{\boldsymbol{i,t}}, \boldsymbol{p}_{\bar{vi},\boldsymbol{t}}\right) \\ &\quad = p_{i,t}^{M} T_{i,t}^{M}\left(p_{i,t}^{M}\right) + \left(p_{i,t}^{F} + p_{i,t}^{C}\right) T_{i,t}^{\mathrm{UP}}\left(p_{i,t}^{F} + p_{i,t}^{C}\right) \\ &\qquad + p_{i,t}^{F} T_{F,t}\left(\boldsymbol{p}_{\boldsymbol{i,t}}, \boldsymbol{p}_{\bar{\boldsymbol{i}},\boldsymbol{t}}\right) + p_{i,t}^{C} T_{C,t}\left(p_{i,t}^{C}\right) \\ &\quad = p_{i,t}^{M} T_{i,t}^{M}\left(p_{i,t}^{M}\right) + p_{i,t}^{F}\left[T_{i,t}^{\mathrm{UP}}\left(p_{i,t}^{F} + p_{i,t}^{C}\right) + T_{F,t}\left(\boldsymbol{p}_{\boldsymbol{i,t}}, \boldsymbol{p}_{\bar{\boldsymbol{i}},\boldsymbol{t}}\right)\right] \\ &\qquad + p_{i,t}^{C}\left[T_{i,t}^{\mathrm{UP}}\left(p_{i,t}^{F} + p_{i,t}^{C}\right) + T_{C,t}\left(p_{i,t}^{C}\right)\right] \end{aligned} \tag{12}$$

where $\boldsymbol{p}_{\boldsymbol{i,t}} = (p_{i,t}^{M}, p_{i,t}^{F}, p_{i,t}^{C}, p_{i,t}^{D})$ is the strategy vector of MD i at time slot t; $\boldsymbol{p}_{\bar{\boldsymbol{i}},\boldsymbol{t}}$ is the vector formed by the strategies of all MDs except the ith one at time slot t, which can be denoted as $\boldsymbol{p}_{\bar{\boldsymbol{i}},\boldsymbol{t}} = \{\ldots, p_{i-1,t}^{M}, p_{i-1,t}^{F}, p_{i-1,t}^{C}, p_{i-1,t}^{D}, p_{i+1,t}^{M}, p_{i+1,t}^{F}, p_{i+1,t}^{C}, p_{i+1,t}^{D} \ldots\}$.

Nevertheless, some of the requests may not be executed but have to be dropped, e.g., due to energy shortage for local computing or offloading to the cloud, meanwhile when the wireless channel from MDs to the fog node is in deep fading, the data of the requests cannot be successfully delivered. To take this aspect into consideration, we penalize per dropped task by cost μ_i, thus the punishment cost for MD i at time slot t can be expressed as follows:

$$C_{i,t} = \mu_i p_{i,t}^{D} \lambda_{i,t} \tau. \tag{13}$$

Consequently, the execution cost for MD i at time slot t, can be formulated as follows:

$$\begin{aligned} &EC_{i,t}\left(\boldsymbol{p}_{\boldsymbol{i,t}}, \boldsymbol{p}_{\bar{\boldsymbol{i}},\boldsymbol{t}}\right) \\ &\quad = T_{i,t}\left(\boldsymbol{p}_{\boldsymbol{i,t}}, \boldsymbol{p}_{\bar{\boldsymbol{i}},\boldsymbol{t}}\right) + \bar{\alpha} C_{i,t}\left(p_{i,t}^{D}\right) \\ &\quad = \frac{p_{i,t}^{M}}{u_i^{M}\left(1 - l_{i,t}^{M}\right) - \lambda_{i,t} p_{i,t}^{M}} + \frac{\left(p_{i,t}^{F} + p_{i,t}^{C}\right)^2 \lambda_{i,t} \tau \theta_i}{W\log_2\left(1 + \frac{q_{i,t}g_{i,t}^{BS}}{w_{i,t} + \sum_{j\in N, j\neq i} q_{j,t}g_{j,t}^{BS}}\right)} \\ &\qquad + p_{i,t}^{F} \frac{2u^{F}\left(1 - l_t^{F}\right) - \sum_{i=1}^{N} \lambda_{i,t} p_{i,t}^{F}/c}{2u^{F}\left(1 - l_t^{F}\right)\left[u^{F}\left(1 - l_t^{F}\right) - \sum_{i=1}^{N} \lambda_{i,t} p_{i,t}^{F}/c\right]} \\ &\qquad + p_{i,t}^{C}\left(T_{FC} + \frac{1}{u^{C}}\right) + \bar{\alpha} \mu_i p_{i,t}^{D} \lambda_{i,t} \tau \end{aligned} \tag{14}$$

where $\bar{\alpha}$ is the weight of task dropping cost.

Due to the existence of social relationships, it is natural that users would take into account the effect of its neighbors' decision. So users are coupled in the social domain due to the social ties among them, and MD i aims to choose the strategy $\boldsymbol{p}_{\boldsymbol{i,t}} = (p_{i,t}^{M}, p_{i,t}^{F}, p_{i,t}^{C}, p_{i,t}^{D})$ to minimize its social group execution cost, defined as

$$\mathrm{SEC}_{i,t}\left(\boldsymbol{p}_{\boldsymbol{i,t}}, \boldsymbol{p}_{\bar{\boldsymbol{i}},\boldsymbol{t}}\right) \triangleq EC_{i,t}\left(\boldsymbol{p}_{\boldsymbol{i,t}}, \boldsymbol{p}_{\bar{\boldsymbol{i}},\boldsymbol{t}}\right) + \sum_{j\in N_i^S} s_{ij} EC_{j,t}\left(\boldsymbol{p}_{\boldsymbol{j,t}}\right). \tag{15}$$

So the average social group execution cost of MD i during a period of time can be denoted as follows:

$$\mathrm{MSEC}_i\left(\boldsymbol{p_i}, \boldsymbol{p_i^-}\right) = \frac{1}{T}\sum_{t=0}^{T-1}\mathrm{SEC}_{i,t}\left(\boldsymbol{p_{i,t}}, \boldsymbol{p_{i,t}^-}\right) \tag{16}$$

where $\boldsymbol{p_i} = [\boldsymbol{p_{i,0}}, \boldsymbol{p_{i,1}}, \ldots, \boldsymbol{p_{i,t}}, \ldots]$ is the strategies vector of MD i of all the time slots; $\boldsymbol{p_i^-}$ is the strategies vector of all MDs other than MD i of all the time slots, which can be denoted as $\boldsymbol{p_i^-} = \{\ldots, \boldsymbol{p_{i-1,0}}, \boldsymbol{p_{i-1,1}}, \ldots, \boldsymbol{p_{i-1,T}}, \boldsymbol{p_{i+1,0}}, \boldsymbol{p_{i+1,1}}, \ldots, \boldsymbol{p_{i+1,T}}, \ldots\}$.

We next consider the distributed decision making problem among the MDs for making their average social group execution cost minimize. We formulate the problem as follows:

$$\min_{\boldsymbol{p_i}} \quad \mathrm{MSEC}_i\left(\boldsymbol{p_i}, \boldsymbol{p_i^-}\right). \tag{17}$$

Subject to

$$\sum_{i=1}^{N} p_{i,t}^F \lambda_{i,t} - cu^F\left(1 - l_t^F\right) < 0 \tag{18a}$$

$$\lambda_{i,t} p_{i,t}^M - u_i^M\left(1 - l_{i,t}^M\right) < 0 \tag{18b}$$

$$p_{i,t}^M + p_{i,t}^F + p_{i,t}^C + p_{i,t}^D = 1 \tag{18c}$$

$$0 \le p_{i,t}^M, p_{i,t}^F, p_{i,t}^C, p_{i,t}^D \le 1 \tag{18d}$$

$$x_i \kappa_i p_{i,t}^M \lambda_{i,t} \tau \theta_i + \frac{q_{i,t}\left(p_{i,t}^F + p_{i,t}^C\right)\lambda_{i,t}\tau\theta_i}{v_{i,t}} \le B_{i,t} \tag{18e}$$

$$B_{i,t+1} = B_{i,t} - E_{i,t} + e_{i,t} \tag{18f}$$

$$\forall i \in \mathcal{N}, t \in \mathcal{T} \tag{18g}$$

where $v_{i,t} = W\log_2(1 + [(q_{i,t}g_{i,t}^{BS})/(w_{i,t} + \sum_{j\in N, j\neq i} q_{j,t}g_{j,t}^{BS})])$. Constraint (18a) is derived from (7). It shows that the request arrival rate at each server should not exceed the service rate to ensure a stable queue. Nevertheless, due to the energy causality constraints (18d) and (18e), the MDs' decisions are coupled among different time slots, which makes the problem difficult to be tackled. From [13], we can find that by introducing a nonzero lower bound, $E_{i,t}^{\min}$, and a reasonable upper bound $E_{i,t}^{\max}$, on the battery at each time slot, such coupling effect can be eliminated and the system operation can be optimized by ignoring energy constraints (18d) and (18e). Thus, we introduce a modified version of the above problem as follows:

$$\min_{\boldsymbol{p_i}} \quad \mathrm{MSEC}_i\left(\boldsymbol{p_i}, \boldsymbol{p_i^-}\right)$$

Subject to

$$(18\mathrm{a}) - (18\mathrm{c}),\ (18\mathrm{g}) \tag{19a}$$

$$E_{i,t} \in \{0\} \cup \left[E_{i,t}^{\min}, E_{i,t}^{\max}\right]. \tag{19b}$$

In the following, we propose a game theoretic approach in order to achieve efficient computation offloading decision makings among the MDs. Game theory is a powerful tool to analyze the interactions among multiple users who focus on their own interests. A Nash equilibrium has the nice self-stability property that all the users can achieve a mutually satisfactory solution and no user has the incentive to deviate unilaterally. Moreover, by using the intelligence of each individual MD user, game theory is a useful framework for devising decentralized mechanisms with low complexity, such that users can self-organize into a mutually satisfactory solution. It can ease the heavy burden of complex centralized management and reduce the communicating overhead between the fog and MDs.

In this setting, the decisions of all MDs are mutually dependent and the proposed model is a GNEP. The GNEP differs from classical Nash equilibrium problem (NEP) in that, while in a NEP, only the players' objective functions depend on the other players' strategies, but in a GNEP both the objective functions and the strategy sets depend on the other players' strategies. In our problem, the dependence of each player strategy set on the other players' strategies is represented by the constraint (18a), which includes all the MDs' decision variables. Then we will introduce a lemma to illustrate the proposed problem is a jointly convex GNEP so that we can use exponential penalty function and semi-smooth Newton method proposed in the following section to address it effectively.

Lemma 1: The derived GNEP is a jointly convex GNEP.

Proof: See the Appendix. ■

V. Proposed Solution

In this section, as we have proved that the derived GNEP is a jointly convex problem, we can use exponential penalty function method to solve the GNEP above. We reformulate the GNEP as a sequence of smoothing penalized NEPs by means of a partial penalization of the coupling constraints where the exponential penalty functions are used. Furthermore, we formulate the KKT conditions for smoothing penalized NEPs into a system of nonsmooth equations, and then apply the semi-smooth Newton method with Armijo line search to solve the system. The specific algorithm is shown below.

By careful observation, we can find that (18a) is a coupled constraint, which includes other participants' decision variables. Other constraints, such as (18b), (19b) only depend on the MD i itself. As we all know, solving a classic NEP is much easier than solving a GNEP. In this section, through partial punishing the difficult coupling constraints in GNEP, which helps us to solve a classical NEP instead of solving a more difficult GNEP.

With punishing the coupling constraints with exponential penalty function, the original problem is changed as follows:

$$\min_{\boldsymbol{p_i}} \quad \mathrm{MSEC}_i\left(\boldsymbol{p_i}, \boldsymbol{p_i^-}\right) + \frac{1}{\rho}\sum_{t=0}^{T-1}\exp\left[\rho g_{i,t}\left(\boldsymbol{p_{i,t}}, \boldsymbol{p_{i,t}^-}\right)\right] \tag{20}$$

s.t.

$$\lambda_{i,t} p_{i,t}^M - u_i^M\left(1 - l_{i,t}^M\right) < 0 \tag{21a}$$

$$x_i \kappa_i p_{i,t}^M \lambda_{i,t} \tau \theta_i + \frac{q_{i,t}\left(p_{i,t}^F + p_{i,t}^C\right)\lambda_{i,t}\tau\theta_i}{v_{i,t}} - E_{i,t}^{\max} \le 0 \tag{21b}$$

$$E_{i,t}^{\min} - x_i \kappa_i p_{i,t}^M \lambda_{i,t} \tau \theta_i - \frac{q_{i,t}\left(p_{i,t}^F + p_{i,t}^C\right)\lambda_{i,t}\tau\theta_i}{v_{i,t}} \le 0 \tag{21c}$$

$$p_{i,t}^M + p_{i,t}^F + p_{i,t}^C + p_{i,t}^D = 1 \quad \forall t \in \mathcal{T} \tag{21d}$$

where $g_{i,t}(\boldsymbol{p_{i,t}}, \boldsymbol{p_{i,t}^-})$ is the short description of $\sum_{i=1}^{N} p_{i,t}^F \lambda_{i,t} - cu^F(1 - l_t^F)$. Here, we relax the constraints as we neglect such a constraint denoted by $0 \leq p_{i,t}^{M}, p_{i,t}^{F}, p_{i,t}^{C}, p_{i,t}^{D} \leq 1$. Nevertheless, the relaxation has no effect on the optimal strategy vector as we can autonomously select the optimal ones ranged on [0, 1] at last.

Then the convergence theorem of the exponential penalty function method is given in the following theorem.

Theorem 1: Let $\{\rho^k\}$ be a positive sequence that tends to be infinite. For each k, $\boldsymbol{p}_i^k$ is the solution of the NEP when $\rho = \rho^k$. Set $\bar{\boldsymbol{p}}_i$ is a cluster point of the sequence $\{\boldsymbol{p}_i^k\}$, and satisfies the inequality constraints, thus $\bar{\boldsymbol{p}}_i$ is the solution of the GNEP.

Proof: See [24, Th. 1]. ■

So the original GNEPs are evolved to a list of classical NEPs. From Theorem 1, we can find that solving a list of NEPs can obtain the solution of GNEP. For the sake of simplicity, we use $h_{i,t}^{(k)}(\boldsymbol{p_{i,t}}) \leq 0 \quad (k = 1, 2, 3, 4)$ to replace the constraint (21a)–(21d), respectively. So the derived classical NEP denoted in a simple form is expressed as follows:

$$\begin{aligned} \min_{\boldsymbol{p_i}} \quad & \mathrm{MSE}C_i\big(\boldsymbol{p_i}, \boldsymbol{p_i^-}\big) + \frac{1}{\rho}\sum_{t=0}^{T-1} \exp\big[\rho g_{i,t}\big(\boldsymbol{p_{i,t}}, \boldsymbol{p_{i,t}^-}\big)\big] \\ \text{s.t.} \quad & h_{i,t}^{(k)}\big(p_{i,t}\big) \leq 0, k = 1, 2, 3, 4; t = 0, \ldots, T-1. \end{aligned} \tag{22}$$

The KKT condition for the NEP displayed in (22) can be denoted as follows:

$$\begin{aligned} &\nabla_{\boldsymbol{p_i}} \mathrm{MSE}C_i\big(\boldsymbol{p_i}, \boldsymbol{p_i^-}\big) + \sum_{t=0}^{T-1} \exp\big[\rho g_{i,t}(\boldsymbol{p_i})\big] \nabla_{\boldsymbol{p_i}} g_{i,t}\big(\boldsymbol{p_i}, \boldsymbol{p_i^-}\big) \\ &\quad + \sum_{t=0}^{T-1}\sum_{k=1}^{4} \beta_{i,t}^{(k)} \nabla_{\boldsymbol{p_i}}\Big(h_{i,t}^{(k)}\Big) = 0. \end{aligned} \tag{23}$$

Obviously, assemble all of the systems for $i = 1, 2, \ldots, N$, we can obtain the following equivalent system:

$$L(\boldsymbol{p}, \boldsymbol{\beta}) = \left\{ \begin{array}{l} \nabla_{\boldsymbol{p_i}} \mathrm{MSE}C_i\big(\boldsymbol{p_i}, \boldsymbol{p_i^-}\big) + \sum\limits_{t=0}^{T-1} \exp\big[\rho g_{i,t}(\boldsymbol{p_i})\big] \\ \nabla_{\boldsymbol{p_i}} g_{i,t}(\boldsymbol{p_i}) + \sum\limits_{t=0}^{T-1}\sum\limits_{k=1}^{4} \beta_{i,t}^{(k)} \nabla_{\boldsymbol{p_i}}\Big(h_{i,t}^{(k)}\Big) \end{array} \right\}_{i=1}^{N} = \boldsymbol{0} \tag{24}$$

where $\boldsymbol{p} = [\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_i} \ldots, \boldsymbol{p_N}]$ is all the MDs' strategy vector, and $\boldsymbol{\beta} = (\beta_{i,t}^{(k)})^T, k = 1, 2, 3, 4; \ i = 1, 2, \ldots N; t = 1, 2, \ldots T$ is the coefficients of KKT condition for all the MDs during the time slots. We can see that $L(\boldsymbol{p}, \boldsymbol{\beta})$ is a zero vector of large dimension. To solve $\boldsymbol{p}$ and $\boldsymbol{\beta}$, we introduce Fischer–Burmeister function $\varphi(a, b) := \sqrt{a^2 + b^2} - (a + b)$, and construct the following form:

$$\Phi(\boldsymbol{\omega}) = \Phi(\boldsymbol{p}, \boldsymbol{\beta}) = \begin{pmatrix} L(\boldsymbol{p}, \boldsymbol{\beta}) \\ \phi(H(\boldsymbol{p}), \boldsymbol{\beta}) \end{pmatrix} = 0 \tag{25}$$

where $\phi(H(\boldsymbol{p}), \boldsymbol{\beta}) := (\ldots, \varphi(-h_{i,t}^{(k)}, \beta_{i,t}^{(k)}), \ldots)^T$.

We use the semi-smooth Newton method to solve the system $\Phi(\boldsymbol{\omega}) = 0$, which is equal to solving problem (20). First, we define the value function $\Psi(\boldsymbol{\omega}) := (1/2)\Phi^T(\boldsymbol{\omega})\Phi(\boldsymbol{\omega})$, then semi-smooth Newton method is described as Algorithm 1. At last, we choose the optimal solution on interval (0, 1) from the obtained solutions from Algorithm 1.

TABLE II
SIMULATION PARAMETERS

Parameters (Units)	MD 1	MD 2	MD 3
u_i^M (MIPS)	1.6	1.8	2.0
$l_{i,t}^M$	0.10	0.30	0.25
θ_i (bits)	8e+6	8e+6	8e+6
κ_i (joule/cycle)	0.1	0.1	0.1
x_i (cycles)	500	600	700
$E_{i,t}^{max}$ (joules)	10	10	10
$E_{i,t}^{min}$ (joules)	0	0	0

Algorithm 1 Proposed Semi-Smooth Newton Algorithm

```
Step 0: Initialization
Setting initial point: $\boldsymbol{\omega}^0 = \big(\boldsymbol{p}^0, \boldsymbol{\beta}^0\big)$, $\rho > 2$, $\sigma \in \big(0, \frac{1}{2}\big)$, $\alpha \in (0, 1)$, $\varepsilon \geq 0$, $k1 \in (0, 1)$, $p1 > 2$,let $k = 0$.
Step 1: Termination determination
if $\big\|\Psi\big(\boldsymbol{\omega}^k\big)\big\| \leq \varepsilon$ then
    Return $\boldsymbol{\omega}^k = \big(\boldsymbol{p}^k, \boldsymbol{\beta}^k\big)$;
else
    Go to step 2;
end if
Step 2: Direction generation
Choosing $H_k \in \partial\Phi\big(\boldsymbol{\omega}^k\big)$, solving the value of $\boldsymbol{d}^k$ the system $H_k\boldsymbol{d}^k = -\Phi\big(\boldsymbol{\omega}^k\big)$
if $\boldsymbol{d}^k$ can't be obtained or does not satisfy the following condition: $\nabla\Psi\big(\boldsymbol{\omega}^k\big)^T\boldsymbol{d}^k \leq -k1\big\|\boldsymbol{d^k}\big\|^{p1}$ then
    Setting $\boldsymbol{d^k} = -\nabla\Psi\big(\boldsymbol{\omega}^k\big)$;
end if
Step 3: Armijo linear search
Searching the smallest nonnegative integer $m^k$ which satisfies the following inequality: $\Psi\big(\boldsymbol{\omega}^k + \alpha^{m_k}\boldsymbol{d}^k\big) \leq \Psi\big(\boldsymbol{\omega}^k\big) + \sigma\alpha^{m_k}\nabla\Psi\big(\boldsymbol{\omega}^k\big)^T\boldsymbol{d}^k$.
Step 4: Rectification
Setting $\boldsymbol{\omega}^{k+1} = \boldsymbol{\omega}^k + \alpha^{m_k}\boldsymbol{d}^k$, $k = k + 1$, return to Step 1.
```

We also show the convergence of the semi-smooth Newton algorithm in the following proposition.

Proposition 1: The semi-smooth Newton algorithm, with strong system computing power, inherits many excellent features from the classic Newton algorithm. Through determining the step size of the Newton direction with the linear search strategy, it avoids the sensitivity of the algorithm to initial values. Thus the local convergence becomes global convergence.

Proof: See [25, Sec. 5]. ■

VI. PERFORMANCE EVALUATIONS

In this section, extensive simulations are conducted to illustrate the effectiveness of the proposed algorithm for the GNEP. The parameters of the MDs are given as follows in Table II. For simplify, we assume the fog node consists of ten servers and the service rate of each server is 4 (MIPS), and the workload of each server is $l_t^F = 0.5$. The transmission time from the fog to the central cloud is $T^{FC} = 0.5$ (Second), and the service rate of the central cloud is $u^C = 10$ (MIPS).

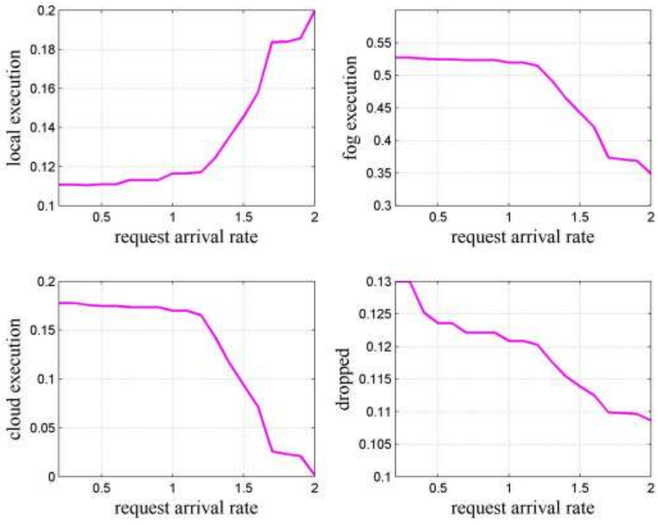

Fig. 2. Optimal execution proportion under different request arrival rates.

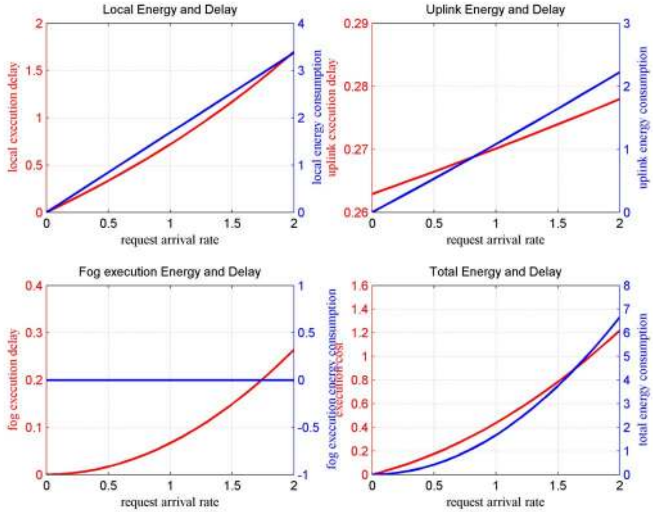

Fig. 3. Execution delay and energy consumption under different request arrival rates.

The social ties among the three MDs are denoted as a matrix, which is $\begin{bmatrix} 1 & 0.25 & 0.80 \\ 0.50 & 1 & 0.65 \\ 0.55 & 0.90 & 1 \end{bmatrix}$. For this matrix, as we have emphasized in Section III-E, the social tie to each MD itself is $s_{ii} = 1$. So we can see that the diagonal elements of the matrix are all 1. Other elements denote the strength of social tie between MD i and MD j, whose values are in interval $(0, 1)$. We can also find that it is a nonsymmetric matrix, in other words, the closeness between two MDs is not identical, which is consistent with the relationship between people in our life. For example, the strength of social tie for MD 1 to MD 2 is 0.25, but the strength of social tie for MD 2 to MD 1 is 0.50.

First, we consider the case that there is only one time slot. In this case, we investigate the optimal decision strategy for locally, fog, central cloud execution and dropped out with request arrival rate increasing for MD 2 in Fig. 2. We can find that with arrival rate increasing, the proportion of local execution first increases slowly at interval $[0, 1]$, and then increases more faster at interval $[1, 2]$. On the contrary, the proportion of fog execution and central cloud execution decrease slowly at interval $[0, 1]$, and decrease quickly at interval $[1, 2]$. This is because that the MD prefers to offload some requests to the fog when the fog have extra capacity to execute the requests. With all MDs' requests pooling to the fog node with limited resource, each MD has to reduce the proportion of offloading to fog. With the interference from other MDs in the uplink transmission, it would cost quite a lot of time and energy to transmit the requests, so the proportion of central cloud execution is also reduced. With the requests arrival rate increasing, the dropped proportion is also reduced.

In Fig. 3, we also investigate the impact of request arrival rate on execution delay and energy consumption for local execution, uplink transmission, fog execution, and also the total process. It needs to be emphasized that we neglect the energy consumption of the MD when the requests are executed in the fog, so we can find a straight line in subplot 3. Generally, a larger request arrival rate can result in a larger execution delay and energy consumption, which can be easily found in any curve in Fig. 3. Additionally, with the request arrival rate increasing, the consumed energy is also increasing, so we must carefully design the offloading strategy to make the execution cost minimize while meeting the energy supply.

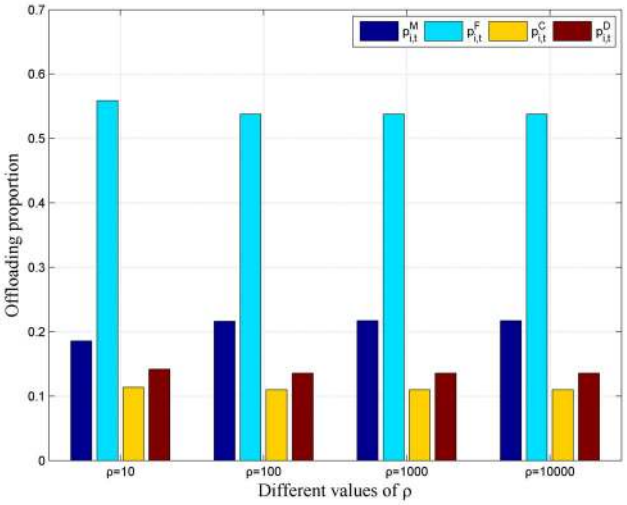

Fig. 4. Effect of ρ on offloading decision.

Additionally, we also compare the decision strategy for MD 2 under a certain request arrival rate for different values of ρ. When $\rho = 10$, $[p^M_{2,t}, p^F_{2,t}, p^C_{2,t}, p^D_{2,t}] = [0.1859, 0.5586, 0.1135, 0.1420]$; when $\rho = 100$, the values are $[p^M_{2,t}, p^F_{2,t}, p^C_{2,t}, p^D_{2,t}] = [0.2164, 0.5378, 0.1100, 0.1358]$; when $\rho = 1000$, the values are $[p^M_{2,t}, p^F_{2,t}, p^C_{2,t}, p^D_{2,t}] = [0.2170, 0.5375, 0.1098, 0.1357]$; when $\rho = 10000$, the values are $[p^M_{2,t}, p^F_{2,t}, p^C_{2,t}, p^D_{2,t}] = [0.2171, 0.5375, 0.1098, 0.1356]$. The numerical results are displayed in Fig. 4. We can see that a larger value of ρ can get a better solution, and the solution would slowly converge to the optimal one with the value of ρ increasing.

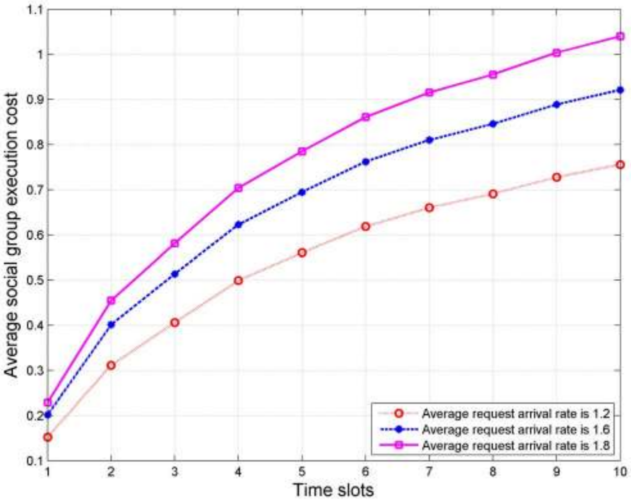

Fig. 5. Execution cost among different time slots under different request arrival rates.

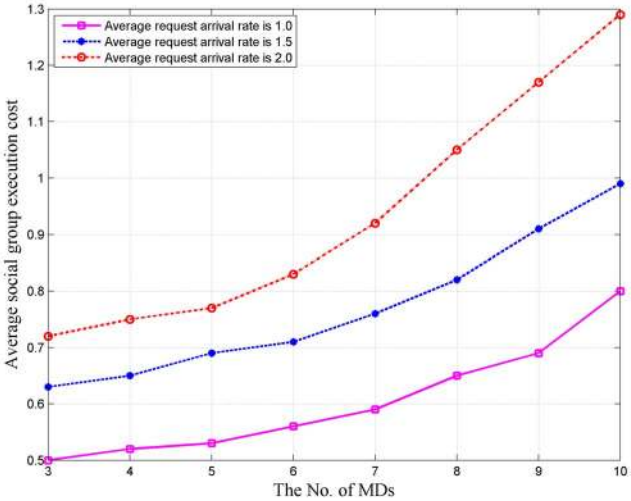

Fig. 6. Average execution cost under different request arrival rates.

Then we focus on a complex situation where the system consists of different time slots. First, we investigate the execution cost among time slots under different request arrival rate, while we assume that in each time slot the harvested energy is identical. It is natural that at the end of each time slot, the remaining energy becomes less, so the MD has to execute more requests locally which would resulting in a larger delay, which means a larger execution cost. We can find this rule from Fig. 5. Typically, a larger request arrival rate cause a larger execution cost, which can be observed by comparing the three curves in Fig. 5.

Next, we also study the execution cost with the number of MDs under different request arrival rates, which is displayed in Fig. 6. We can see that the execution costs are also increasing when the number of MDs increases, which means that the execution delay or the punishment cost become larger. As more and more users compete for resources in the fog with each other, thus intensified the channel interference, reducing the channel transmission rate. So the MD has to execute the request locally or drop them, which leads to a larger execution cost.

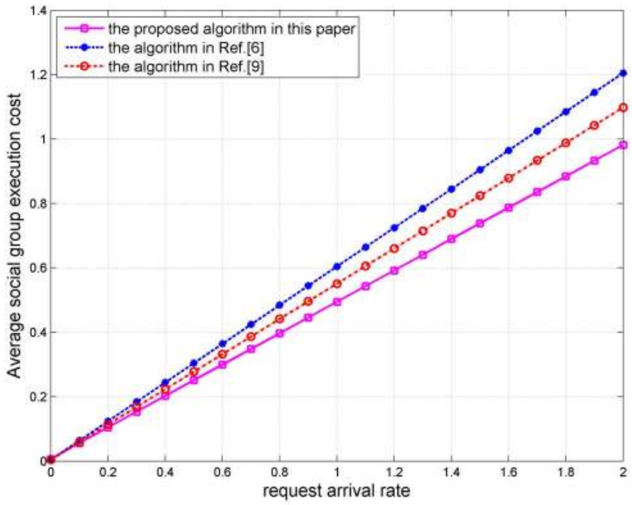

Fig. 7. Performance comparison.

In Fig. 7, we compare our algorithm with the other existing schemes such as the "successive convex approximation method" proposed in [6] and the "Lyapunov optimization-based dynamic computation offloading Algorithm" proposed in [9]. By varying the arrival rates, we can find that our proposed algorithm has a better system performance. This is due to the fact that with the proposed scheme, the GNE point can be achieves, which is a mutually satisfactory solution and no one has the incentive to deviate.

VII. Conclusion

In this paper, a computation offloading problem in a fog computing system has been investigated. We derive the analytic results of energy consumption, delay performance, and cost with assumption of three different queue models at mobile devices, fog, and central cloud. By leveraging the obtained results, we take the social relationships of the EH MDs into the design of offloading scheme. With the objective to minimize the social group execution cost, we advocate game theoretic approach and propose a dynamic computation offloading scheme. Specifically, we formulate a GNEP with various constraints and addressed it by using exponential penalty function method and semi-smooth Newton method with Armijo line search. Extensive performance evaluations are presented to illustrate the effectiveness of the proposed scheme.

Appendix

Proof: First, we will prove that the payoff functions are concave. By derivation, we can obtain the expressions as follows:

$$\begin{aligned}\frac{\partial MESC_i}{\partial p^M_{i,t}} &= \frac{\partial\left[\frac{p^M_{i,t}}{u^M_i\left(1-l^M_{i,t}\right)-\lambda_{i,t}p^M_{i,t}}\right]}{\partial p^M_{i,t}}\\ &= \frac{u^M_i\left(1-l^M_{i,t}\right)}{\left[u^M_i\left(1-l^M_{i,t}\right)-\lambda_{i,t}p^M_{i,t}\right]^2}.\end{aligned} \qquad (26)$$

So we have

$$\frac{\partial^2 MESC_i}{\partial\left(p_{i,t}^M\right)^2} = \frac{2\lambda_{i,t} u_i^M\left(1 - l_{i,t}^M\right)}{\left[u_i^M\left(1 - l_{i,t}^M\right) - \lambda_{i,t} p_{i,t}^M\right]^3} > 0. \tag{27}$$

Simplify the expression by approximation, we can obtain the first-order derivative about variable $p_{i,t}^M$ as follows:

$$\frac{\partial MESC_i}{\partial p_{i,t}^F} \approx \frac{2\left(p_{i,t}^F + p_{i,t}^C\right)}{W v_{i,t}} + 1. \tag{28}$$

So we have

$$\frac{\partial^2 MESC_i}{\partial\left(p_{i,t}^F\right)^2} = \frac{2}{W v_{i,t}} > 0. \tag{29}$$

Here $v_{i,t}$ denotes $\log_2(1 + [(q_{i,t} g_{i,t}^{BS})/(\omega_{i,t} + \sum_{j \in N, j \neq i} q_{j,t} g_{j,t}^{BS})])$.

Similarly, we have

$$\frac{\partial MESC_i}{\partial p_{i,t}^C} = T_{FC} + \frac{1}{u^C} \tag{30}$$

$$\frac{\partial^2 MESC_i}{\partial\left(p_{i,t}^C\right)^2} = 0 \tag{31}$$

$$\frac{\partial MESC_i}{\partial p_{i,t}^D} = \bar{\alpha}\mu_i\lambda_{i,t}\tau \tag{32}$$

$$\frac{\partial^2 MESC_i}{\partial\left(p_{i,t}^D\right)^2} = 0. \tag{33}$$

Combining (27), (29), (31), and (33), we have proved that the payoff functions are concave in its own variable. By observing, we can find (21a)–(21d) are only linear, so the set of constraints are convex apparently. So the derived GNEP is that it satisfies the convexity assumption. Refer to [26], we can draw that the objective formulated problem is a jointly convex GNEP. ■

References

[1] F. Liu *et al.*, "Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 14–22, Jun. 2013.

[2] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3317–3329, Dec. 2015.

[3] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 369–392, 1st Quart., 2014.

[4] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the Internet of Things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, Aug. 2016.

[5] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016, doi: 10.1109/JIOT.2016.2565516.

[6] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[7] L. Jiang *et al.*, "Social-aware energy harvesting device-to-device communications in 5G networks," *IEEE Wireless Commun.*, vol. 23, no. 4, pp. 20–27, Aug. 2016.

[8] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, May 2016.

[9] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[10] T. K. Thuc, E. Hossain, and H. Tabassum, "Downlink power control in two-tier cellular networks with energy-harvesting small cells as stochastic games," *IEEE Trans. Commun.*, vol. 63, no. 12, pp. 5267–5282, Dec. 2015.

[11] H. H. Yang, J. Lee, and T. Q. S. Quek, "Heterogeneous cellular network with energy harvesting-based D2D communication," *IEEE Trans. Wireless Commun.*, vol. 15, no. 2, pp. 1406–1419, Feb. 2016.

[12] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 81–93, Jan. 2015.

[13] W. Zhang *et al.*, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[14] L. Tang, X. Chen, and S. He, "When social network meets mobile cloud: A social group utility approach for optimizing computation offloading in cloudlet," *IEEE Access*, vol. 4, pp. 5868–5879, 2016.

[15] X. Chen, X. Gong, L. Yang, and J. Zhang, "Exploiting social tie structure for cooperative wireless networking: A social group utility maximization framework," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3593–3606, Dec. 2016, doi: 10.1109/TNET.2016.2530070.

[16] Y. Zhang *et al.*, "Social network aware device-to-device communication in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 177–190, Jan. 2015.

[17] C. Xu, C. Gao, Z. Zhou, Z. Chang, and Y. Jia, "Social network-based content delivery in device-to-device underlay cellular networks using matching theory," *IEEE Access*, vol. 5, pp. 924–937, 2016.

[18] D. Hussein, S. N. Han, G. M. Lee, and N. Crespi, "Social cloud-based cognitive reasoning for task-oriented recommendation," *IEEE Cloud Comput.*, vol. 2, no. 6, pp. 10–19, Nov./Dec. 2015.

[19] Z. Ning, F. Xia, X. Kong, and Z. Chen, "Social-oriented resource management in cloud-based mobile networks," *IEEE Cloud Comput.*, vol. 3, no. 4, pp. 24–31, Jul./Aug. 2016.

[20] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct./Dec. 2017, doi: 10.1109/TCC.2015.2449834.

[21] V. Cardellini *et al.*, "A game-theoretic approach to computation offloading in mobile cloud computing," *Math. Program.*, vol. 157, no. 2, pp. 421–449, 2016.

[22] A. Lazar, "The throughput time delay function of an M/M/1 queue (Corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 6, pp. 914–918, Nov. 1983.

[23] R. E. Machol, "Queue theory," *IRE Trans. Educ.*, vol. E-5, no. 2, pp. 99–105, Nov. 2007.

[24] J.-X. Xu, J. Hou, Y.-H. Tan, and E.-M. Feng, "Exponential penalty function method for generalized Nash equilibrium problem," *Oper. Res. Manag. Sci.*, vol. 24, no. 1, pp. 81–88, Feb. 2015.

[25] L. Xie, G. Lütai, and Q. Lian, "The general convex smoothing problem solved by a semismooth Newton algorithm," *Math. Numerica Sinica*, vol. 27, no. 3, pp. 257–266, Aug. 2005.

[26] F. Facchinei and C. Kanzow, "Generalized Nash equilibrium problems," *Ann. Oper. Res.*, vol. 175, no. 1, pp. 177–211, Mar. 2010.