A Strategic Game for Task Offloading among Capacitated UAV-mounted Cloudlets

Weibin Ma
Department of Computer and Information Sciences
University of Delaware
Newark, Delaware, USA
Email: weibinma@udel.edu

Xuanzhang Liu
Department of Computer and Information Sciences
University of Delaware
Newark, Delaware, USA
Email: xzliu@udel.edu

Lena Mashayekhy
Department of Computer and Information Sciences
University of Delaware
Newark, Delaware, USA
Email: mlena@udel.edu

***Abstract*—In this paper, we focus on the computational offloading problem among unmanned aerial vehicles (UAVs) acting as small flying cloudlets that receive compute-intensive tasks from Internet-of-Things (IoT) devices. Different from existing studies, we consider a network of capacitated UAV-mounted cloudlets (NUMC) covering a region, where each UAV is endowed with limited computational resources and a restricted capacity providing edge computing services to IoT users in that region. UAVs aim to optimize their energy consumption while satisfying quality of services of IoT tasks. We formulate the task offloading problem among UAVs as an integer program, and introduce a novel offloading game to model this problem. We prove the existence of pure-strategy Nash equilibrium of our game, where none of the UAVs has incentive to change its offloading decision. We propose a strategic offloading algorithm to solve our proposed game and find a Nash equilibrium. We evaluate the performance of our proposed algorithm by extensive experiments. The results show that a Nash equilibrium exists in NUMC and a desired system performance is achieved as well.**

***Keywords*-Edge Computing, Unmanned Aerial Vehicles, Internet-of-Things, UAV-mounted Cloudlets, Energy Consumption, Offloading Game, Nash Equilibrium.**

I. Introduction

The growth of Internet of Things (IoT) will continue as users enjoy the convenience of mobility and with the emergence and progress of new technologies such as wearable devices and autonomous vehicles. These smart connected devices have limited computational capabilities due to being restricted by weight, size, battery life, and heat dissipation, and they cannot run complex applications requiring intensive computation (such as embedded speech recognition, real-time face recognition, and augmented reality) through relying on their local resources. Hence, offloading computation is becoming the most promising solution to handle this challenge.

Mobile edge computing (MEC) can be leveraged to bridge the gap between the increasing computational demand of the IoT devices and their limited computational capabilities. MEC has been recently introduced as a new computing paradigm [1] that optimizes cloud computing systems to provide a distributed computing solution at the edge of the network, where mobile users utilize computing resources in their vicinity (e.g., cloudlets). Instead of moving a large amount of raw data from IoT user devices to a distant cloud, MEC enables processing data closer to the end users, where data has been produced locally. IoT users can offload their intensive applications to a cloudlet or a remote cloud for the purpose of reducing energy consumption and execution latency. However, deploying cloudlet infrastructure at the edge of the network in MEC is expensive and may not be feasible in many situations (e.g., disaster situations, emergency rescue, unexpected surge in user demand) and regions with only limited or no infrastructure of wireless access points (e.g., remote rural areas, existing massive obstacles).

For these situations, an emerging method considering Unmanned Aerial Vehicles (UAVs) as computational cloudlets is proposed due to their inherent attributes such as aerial mobility, low operating costs, flexible deployment, and wireless communication ability [2]. UAV-mounted cloudlets can provide the edge computing services and enhance the quality-of-service (QoS). To be more precise, IoTs can offload their intensive tasks to UAV-mounted cloudlets by virtue of the wireless communications between the IoTs and the cloudlets. Application offloading to these UAVs has been investigated in several studies [3], [4]. However, these studies only investigate that single UAV can either process all incoming tasks locally or offload them to a static base station or a distant cloud.

In this paper, we focus on offloading tasks among a swarm of UAVs acting as aerial capacitated cloudlets pooling their computational resources to execute tasks from the ground IoT devices in their coverage. UAVs' goal is to reduce their energy consumption, while guaranteeing QoS for the IoT users. We introduce a novel framework, a network of capacitated UAV-mounted cloudlets (NUMC), to deal with the task offloading problem among UAVs. [1] UAVs can choose other UAVs to offload their tasks for computation or execute the tasks locally based on their own interests. We design an optimal offloading mathematical model, and then propose a novel game theoretic solution and a Strategic

[1]We use the terms UAV and cloudlet interchangeably.

Offloading Algorithm (SOA) to optimize the computation offloading problem in the proposed NUMC. We prove that our proposed game is an exact potential game and achieves a *pure-strategy Nash equilibrium* at which not any UAV can individually change its strategy to reduce its cost. To the best of our knowledge, this is *the first work that addresses the computation offloading problem among a swarm of UAVs via exploiting game theory.*

Challenges and Contributions. The main challenges of computation offloading in NUMC are as follows: IoT users and UAVs have their own interests. Enabling each UAV to attain a mutually satisfactory solution while satisfying desired requirements for IoT tasks, is challenging. Coordinating computation offloading of multiple tasks via multiple UAVs is more complex than that of a single cloud or a UAV, where the decision is whether to offload or not. In NUMC, however, each UAV has several offloading decisions. In order to cope with these challenges, we design a decentralized offloading approach by exploiting game theory, and our contributions include:

- The feasibility of computation migration among a swarm of capacitated UAVs with the objective of minimizing their energy consumption, while satisfying QoS requirements of IoTs is studied. The outcome of this study can be used in extreme situations, where the base stations and the cloud are unavailable.
- A novel capacitated offloading game is proposed considering the incentives of UAVs and IoTs. We prove that our game is an *exact potential game*.
- A decentralized Strategic Offloading Algorithm is proposed, and we prove that the algorithm always admits a *pure-strategy Nash equilibrium* when a *feasible solution* exists.
- The numerical results demonstrate the proposed SOA in NUMC can further improve the system performance compared to non-deterministic offloading and local computing strategy profiles.

II. Related Work

Most prior studies on intensive computation offloading from IoTs to a cloud or a cloudlet aim to optimize energy consumption or response time or both for the users [3], [5], [6]. Only a few studies exploit possibility of having UAVs as flying base stations or even as a part of the cloud computing system. Jeong $et\ al.$ [3] proposed a UAV-based mobile cloud computing system in which a single UAV is endowed with computational capabilities to offer computation service to ground users, and the system aimed to optimize the transmission energy of ground users over the bit allocation and UAV's trajectory. Labidi $et\ al.$ [5] devised online learning approaches and deterministic offline approaches for the energy savings of a single user. Huang $et\ al.$ [6] proposed a dynamic offloading algorithm based on Lyapunov optimization to improve the performance in terms of the energy saving while meeting the application deadline.

A few studies [4], [7]–[11], instead, designed decentralized mechanisms to optimize the offloading problem by exploiting game theory. Chen $et\ al.$ [7] proposed a game theoretic approach for solving the multi-user multi-channel offloading to a cloud. Since computing a centralized optimal solution is NP-hard, a distributed game theoretic approach is adopted for achieving efficient computational offloading. Chen $et\ al.$ [8] again devised a decentralized computation offloading game for both of homogeneous and heterogeneous mobile users. Their proposed formulated game admits a Nash equilibrium, however, the mechanism only considers one offloading strategy, the cloud. Messous $et\ al.$ [4] proposed a game theoretic approach to address the offloading decision making problem for UAVs with three strategies (i.e., local, offloading to a base station, and offloading to the cloud) and proved the existence of a Nash equilibrium. However, the offloading strategies among UAVs and the computation capacity of UAVs were not considered. Ma $et\ al.$ [9] focused on the distributed computation offloading strategy of multiple users to a cloudlet via multiple access points. All of the above-mentioned studies neglected the possibility of offloading among UAVs, and they assumed that each single cloudlet/UAV can either process all incoming tasks locally or offload them to a distant cloud.

III. System Model

In this section, we describe the system model with a set of UAVs acting as computational cloudlets, where their objective is to minimize their energy consumption by offloading their intensive IoT tasks to appropriate UAVs while guaranteeing QoS of IoT tasks.

In NUMC, we consider a set of UAVs $U = \{u_1, \ldots, u_n\}$, where n denotes the number of UAVs. Each UAV u_i has a computation task. Let F_i be the CPU frequency (i.e., CPU cycles per second) of UAV u_i, e_i denotes the energy consumption per CPU cycle of u_i, and its capacity is denoted by u_i^{cap}. UAV's capacity is due to its physical-architecture limitation. In fact, executing too many intensive tasks on a UAV may have detrimental impacts on its battery lifetime and tasks' execution time. Thus, UAV u_i can simultaneously execute u_i^{cap} tasks at most. We define each task k with (C^k, S^k), where C^k represents the number of computational cycles required to obtain the outcome of the task, and S^k denotes the data size of the task. Moreover, each IoT task has its own QoS requirement, and thus we consider that each task k of u_i has a maximum tolerable execution time T_{ik}^{max} (execution latency). We assume each task can be either computed locally on the UAV itself or offloaded to another UAV. Following [12], [13], UAVs use frequency division multiple access (FDMA) that enables each UAV to offload tasks to other UAVs. In this technique, each UAV allocates equal bandwidth channels to other UAVs that offload

tasks to and each channel is assigned to one UAV to avoid the channel interference. Using FDMA, IoT tasks can be migrated to a swarm of UAVs without interference. Similar to previous studies in mobile wireless networks [8], [14], for tractability, we assume that the 3D positions of UAVs remain fixed during each computation offloading process. More details are presented in the following subsections.

A. Task Offloading Problem

Due to limited computational resources and available battery of a UAV, executing tasks on a UAV locally might not be efficient in terms of energy savings at the UAV. To tackle this challenge, we introduce a novel approach in NUMC, where the intensive tasks are transmitted via the FDMA technique among UAVs to minimize their energy consumption guaranteeing QoS of IoT tasks. In this subsection, we present the communication and computation models.

1) Communication Model. FDMA requires that the total bandwidth of UAV u_i is equally allocated to all UAVs offloading tasks to it. As a result, the transmission bandwidth (data rate) of u_i allocated to u_j is computed by:

$$r_{ji} = \frac{B_i}{d_i}, \tag{1}$$

where B_i is the total bandwidth of u_i and d_i represents the total number of UAVs communicating with u_i.

2) Computation Model. We define $x_{ij} \in \{0, 1\}$ as the offloading decision variable as follows:

$$x_{ij} = \begin{cases} 1 & \text{if task } k \text{ is offloaded from } u_i \text{ to } u_j, \\ 0 & \text{otherwise.} \end{cases}$$

This shows whether to offload task k from u_i to u_j or not. Note that, the case $j = i$ means the task k will be executed on u_i locally. Similar to [9], we set that every UAV has a specific amount of computational capabilities, and we assume that the computation capabilities of a UAV are fairly assigned to the tasks running on this UAV. Thus, the computation capabilities of u_i to execute a task locally is represented as:

$$f_i = \frac{F_i}{\sum_{j=1}^{n} x_{ji}} \tag{2}$$

In the following, we introduce the energy consumption and execution time functions by taking into account both computation and data transmission aspects of UAVs.

2.1) Energy Consumption for Local Computing. As a task is executed locally rather than offloaded to other UAVs, the energy consumption only depends on the computational capabilities of the UAV and the total number of tasks offloaded to this UAV. The energy consumption for local computing hence can be computed by:

$$E^k_{i \to i} = C^k e_i \tag{3}$$

2.2) Energy Consumption for Offloading. The total energy consumption for completing task k via offloading (which is offloaded from u_i to u_j) consists of three components: the transmission energy consumption from u_i to u_j, the execution energy consumption at u_j, and the backhaul energy consumption of outcomes of computation from u_j to u_i. Similar to many studies (e.g., [6], [8]), the backhaul energy consumption of the outcomes is omitted in our model. This is due to the fact that for many tasks, such as image processing of face recognition, the size of outcome generally is significantly smaller than that of the input data. As a result, the total energy consumption for completing the task via offloading is given by:

$$E^k_{i \to j} = \frac{S^k P_i}{r_{ij}} + C^k e_j, \tag{4}$$

where P_i is the transmission power of u_i, the value of $\frac{S^k P_i}{r_{ij}}$ denotes the total transmission energy required from u_i to u_j, and $C^k e_j$ represents the total execution energy required at u_j. Thus, the energy consumption cost function for computing task k is defined as:

$$E^k_i = \sum_{j=1, j \neq i}^{n} (\frac{S^k P_i}{r_{ij}} + C^k e_j) x_{ij} + C^k e_i x_{ii} \tag{5}$$

2.3) Total Execution Time. Likewise, the execution time function for computing task k is defined as:

$$T^k_i = \sum_{j=1, j \neq i}^{n} (\frac{S^k}{r_{ij}} + \frac{C^k}{f_j}) x_{ij} + \frac{C^k}{f_i} x_{ii}, \tag{6}$$

where $\frac{S^k}{r_{ij}} + \frac{C^k}{f_j}$ represents the total execution time of u_i's task via offloading to u_j, and $\frac{C^k}{f_i}$ denotes the total execution time of u_i's task via local computing. Thus, if a task is offloaded from u_i to u_j, the required processing time includes its transmission time from u_i to u_j via wireless access and its execution time on u_j.

B. Optimal Task Offloading Model

Minimizing both energy consumption on UAVs and execution time of tasks leads to a complex multi-objective optimization problem. One approach, investigated by [4], [7]–[9], is to exploit a weighted method to measure the importance of two or more objectives. However, the values of energy and processing time cannot be always guaranteed to be in the same numerical magnitude. Due to their largely different magnitudes, a strict normalization is needed, which instead increases the complexity of the objective function. In this situation, a common method is to relax one objective and make it a hard constraint. In this paper, we relax the execution time by restricting it with a predetermined deadline and satisfying it as the execution latency constraint. This will guarantee that UAVs provide the computing services based on QoS requirements of the IoT users. Note that we

allow each task to have a different execution deadline. We formulate the energy optimization problem as an Integer Program (IP), called IP-ENERGY, as follows:

$$\text{Minimize } E^{total} = \sum_{i=1}^{n} E_i^k \tag{7}$$

Subject to:

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall i \in \{1,\dots,n\}, \tag{8}$$

$$T_i^k \le T_{ik}^{max}, \qquad \forall i \in \{1,\dots,n\}, \tag{9}$$

$$\sum_{i=1}^{n} x_{ij} \le u_j^{cap}, \qquad \forall j \in \{1,\dots,n\}, \tag{10}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i,j \in \{1,\dots,n\}. \tag{11}$$

The objective function (7) is to minimize the total energy consumption of all tasks in NUMC. Constraints (8) ensure that each task is executed by only one UAV. Constraints (9) guarantee that processing time of each task is not exceeding its execution deadline T_{ik}^{max}. Constraints (10) ensure that the number of tasks received by u_j including its own task is not higher than its current capacity u_j^{cap}. Constraints (11) guarantee that the decision variables are binary.

To solve IP-ENERGY, we propose a strategic offloading game that will be introduced in the next section.

IV. Capacitated Offloading Game

According to Eq. (1-4), if many UAVs simultaneously select the same UAV to offload their tasks, the data rate will be lower, which in turn involves a higher transmission energy consumption for offloading. Moreover, the execution energy consumption on that UAV for each task will be decreased.

Definitely, each UAV's offloading decision will directly impact other UAVs in NUMC. To optimize the overall system energy, it is important to make the best offloading decision for the task on each UAV. We introduce a game theoretic approach to model our problem using concepts from *exact potential games* [15]. A main reason for selecting the game theory approach to solve this problem is that each UAV can be considered as an individual player with private interests. UAVs need to evaluate offloading strategies in the presence of congestion arising from the decisions made by themselves and other UAVs. A strategic game is one of the most promising solutions for our problem since it is powerful to analyze the interactions among UAVs which act in their own interests. The solution, where no player has incentive to individually change its strategy, is called a *Nash equilibrium* [16].

A. Offloading Game Formulation

According to Eq. (2-3), the total energy consumption of executing task k locally on u_i is:

$$E_{i\to i}^k = C^k e_i = C^k \gamma (\frac{F_i}{\sum_{s=1}^{n} x_{si}})^2, \tag{12}$$

where e_i is approximately linearly proportional to the square of computation frequency [3]. Thus, $e_i = \gamma (f_i)^2$, where γ is the effective switched capacitance of the cloudlet processor, and is set to 1.2×10^{-28}.

The total energy consumption of offloading task k from u_i to u_j is:

$$E_{i\to j}^k = \frac{S^k P_i}{B_j / d_j} + C^k \gamma (\frac{F_j}{\sum_{s=1}^{n} x_{sj}})^2, \qquad i \ne j. \tag{13}$$

We define $y_i \in \{1,\dots,n\}$ as u_i's strategy (decision). Corresponding to other UAVs' decisions, u_i can choose any UAV including itself for the task execution, where $y_i = i$ represents to execute the task on u_i locally and $y_i = j, i \ne j$, denotes to offload the task from u_i to u_j. Moreover, we define $y_{-i} = (y_1,\dots,y_{i-1},y_{i+1},\dots,y_n)$ as the offloading strategies of all UAVs except u_i's. Therefore, the cost function of u_i for task k, denoted as $Z_i^k(y_i, y_{-i})$, considering the feasibility constraints (Eq. (9-10)) is defined as:

$$Z_i^k(y_i, y_{-i}) = \begin{cases} E_{i\to i}^k, & \text{if } y_i = i, \\ E_{i\to j}^k, & \text{if } y_i = j, \quad i \ne j. \end{cases} \tag{14}$$

We now define the task offloading problem as a strategic game, called Capacitated Offloading Game (COG):

Definition 1. A Capacitated Offloading Game is described by $\mathcal{G}\langle \mathbf{U}, \mathbf{Y}, \mathbf{Z}\rangle$, where $\mathbf{U} = \{u_1, u_2, \dots, u_n\}$ denotes the set of UAVs (i.e., players), $\mathbf{Y}$ represents the set of decisions of the UAVs, and $\mathbf{Z}$ is defined as the set of cost functions for the UAVs. The game aims to minimize the cost function $Z_i^k(y_i, y_{-i})$ of every u_i with respect to other UAVs' strategies:

$$\begin{aligned} &\underset{y_i}{\textit{minimize}} \quad Z_i^k(y_i, y_{-i}), \\ &\textit{subject to} \quad y_i \in \{1,\dots,n\}. \end{aligned} \tag{15}$$

We now define the *Nash equilibrium* in COG:

Definition 2. A *Nash equilibrium* of COG $\mathcal{G}\langle \mathbf{U}, \mathbf{Y}, \mathbf{Z}\rangle$ is a strategy profile $\mathbf{y}^* = \{y_1^*, y_2^*, \dots, y_n^*\}$, where no UAV u_i can do better by choosing an action y_i different from y_i^*, given that every other UAV u_j adheres to y_j^*. That is:

$$Z_i(y_i^*, y_{-i}^*) \le Z_i(y_i, y_{-i}^*), \ \ \forall y_i, i \in \{1,\dots,n\}. \tag{16}$$

According to this definition, the Nash equilibrium has a stability property, that is no UAV has incentive to deviate from its offloading decision if it is at the Nash equilibrium. That means every UAV is at its best strategy once a Nash equilibrium is reached. We next prove the existence of a *Nash equilibrium* in our proposed COG.

B. Existence of Nash Equilibrium

In order to prove our proposed capacitated offloading game has a *Nash equilibrium*, we need to show that COG is an *exact potential game* by defining a potential function. According to [15], a potential function Φ can be defined

on possible solutions by showing that any improving move by one of the players to lower its own cost reduces the value of Φ. Since the set of possible solutions is finite, any sequence of improving moves leads to a *pure-strategy Nash equilibrium*.

Definition 3. COG is an exact potential game if and only if a potential function $\Phi(\mathbf{Y})$: $\mathbf{Y} \mapsto \{1, \ldots, n\}$ exists such that, $\forall i \in \{1, \ldots, n\}$:

$$Z_i(y_i, y_{-i}) - Z_i(y_i', y_{-i}) = \Phi(y_i, y_{-i}) - \Phi(y_i', y_{-i}), \quad \forall y_i, y_i' \in \mathbf{Y}_i, \forall y_{-i} \in \mathbf{Y}_{-i}, \tag{17}$$

where $\mathbf{Y}_i$ and $\mathbf{Y}_{-i}$ are the strategy spaces of UAV u_i and other UAVs, respectively.

That means, the change in a single UAV's energy cost due to its own strategy deviation causes exactly the same amount of change in the potential function (i.e., $\Delta Z_i = \Delta\Phi$). We define $\Phi(\mathbf{Y})$ as:

$$\begin{aligned}\Phi(\mathbf{Y}) = &\sum_{i=1}^{n}\left(C^k\gamma(\frac{F_i}{\sum_{j=1}^{n}\mathcal{I}(y_j=i)})^2\mathcal{I}(y_i=i)\right)+\\ &\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\left(\frac{S^kP_i}{B_j}\sum_{s=1,s\neq i}^{n}\mathcal{I}(y_s=j)+\right.\\ &\left.C^k\gamma(\frac{F_j}{\sum_{s=1}^{n}\mathcal{I}(y_s=j)})^2\right)\mathcal{I}(y_i=j),\end{aligned} \tag{18}$$

where function $\mathcal{I}(y_j = i) = 1$ if and only if $y_j = i$, otherwise $\mathcal{I}(y_j = i) = 0$. Thus, $\sum_{j=1,j\neq i}^{n}\mathcal{I}(y_j = i)$ indicates the number of UAVs offloading their tasks to u_i.

Theorem 1. COG $\mathcal{G}\langle$**U, Y, Z**$\rangle$ with potential function $\Phi(\mathbf{Y})$ defined in Eq. (18) is an *exact potential game*.

Proof: In order to prove COG is an exact potential game, we need to prove our potential function $\Phi(\mathbf{Y})$, defined in Eq. (18), is fully applicable in all possible cases as follows:

1) $\forall i, j \in \{1, \ldots, n\}$, if UAV u_i chooses to change the offloading decision from i (i.e., locally) to j, according to Eq. (18), we obtain that:

$$\begin{aligned}&\Phi(i, y_{-i}) - \Phi(j, y_{-i})\\ =&\left[\sum_{i=1}^{n}\left(C_i^k\gamma(\frac{F_i}{\sum_{m=1}^{n}\mathcal{I}(y_m=i)})^2\mathcal{I}(y_i=i)\right)\right]-\\ &\left[\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\left(\frac{S_i^kP_i}{B_j}\sum_{v=1,v\neq i}^{n}\mathcal{I}(y_v=j)+\right.\right.\\ &\left.\left.C_i^k\gamma(\frac{F_j}{\sum_{m=1}^{n}\mathcal{I}(y_m=j)})^2\right)\mathcal{I}(y_i=j)\right]\\ =&\left[C_i^k\gamma(\frac{F_i}{\sum_{m=1}^{n}\mathcal{I}(y_m=i)})^2\right]-\\ &\left[\frac{S_i^kP_i}{B_j}\sum_{v=1}^{n}\mathcal{I}(y_v=j)+C_i^k\gamma(\frac{F_j}{\sum_{m=1}^{n}\mathcal{I}(y_m=j)})^2\right]\\ =&[C_i^k\gamma(f_i)^2]-\left[\frac{S_i^kP_i}{B_j}\sum_{v=1}^{n}\mathcal{I}(y_v=j)+C_i^k\gamma(f_j)^2\right]\\ =&Z_i(i, y_{-i}) - Z_i(j, y_{-i})\end{aligned} \tag{19}$$

2) Similarly, if UAV u_i chooses to change the offloading decision from j to i (i.e., locally), we obtain:

$$\begin{aligned}&\Phi(j, y_{-i}) - \Phi(i, y_{-i})\\ =&\left[\frac{S_i^kP_i}{B_j}\sum_{v=1}^{n}\mathcal{I}(y_v=j)+C_i^k\gamma(\frac{F_j}{\sum_{m=1}^{n}\mathcal{I}(y_m=j)})^2\right]-\\ &\left[C_i^k\gamma(\frac{F_i}{\sum_{m=1}^{n}\mathcal{I}(y_m=i)})^2\right]\\ =&\left[\frac{S_i^kP_i}{B_j}\sum_{v=1}^{n}\mathcal{I}(y_v=j)+C_i^k\gamma(f_j)^2\right]-[C_i^k\gamma(f_i)^2]\\ =&Z_i(j, y_{-i}) - Z_i(i, y_{-i})\end{aligned} \tag{20}$$

3) $\forall i, h, l \in \{1, 2, \ldots, n\}$, if UAV u_i chooses to change the offloading decision from h to l, according to (18), we can compute that:

$$\begin{aligned}&\Phi(h, y_{-i}) - \Phi(l, y_{-i})\\ =&\left[\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\left(\frac{S_i^kP_i}{B_h}\sum_{v=1,v\neq i}^{n}\mathcal{I}(y_v=h)+\right.\right.\\ &\left.\left.C_i^k\gamma(\frac{F_h}{\sum_{m=1}^{n}\mathcal{I}(y_m=h)})^2\right)\mathcal{I}(y_i=h)\right]-\\ &\left[\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\left(\frac{S_i^kP_i}{B_l}\sum_{v=1,v\neq i}^{n}\mathcal{I}(y_v=l)+\right.\right.\\ &\left.\left.C_i^k\gamma(\frac{F_l}{\sum_{m=1}^{n}\mathcal{I}(y_m=l)})^2\right)\mathcal{I}(y_i=l)\right]\\ =&\left[\frac{S_i^kP_i}{B_h}\sum_{v=1}^{n}\mathcal{I}(y_v=h)+C_i^k\gamma(\frac{F_h}{\sum_{m=1}^{n}\mathcal{I}(y_m=h)})^2\right]-\\ &\left[\frac{S_i^kP_i}{B_l}\sum_{v=1}^{n}\mathcal{I}(y_v=l)+C_i^k\gamma(\frac{F_l}{\sum_{m=1}^{n}\mathcal{I}(y_m=l)})^2\right]\\ =&\left[\frac{S_i^kP_i}{B_h}\sum_{v=1}^{n}\mathcal{I}(y_v=h)+C_i^k\gamma(f_h)^2\right]-\\ &\left[\frac{S_i^kP_i}{B_l}\sum_{v=1}^{n}\mathcal{I}(y_v=l)+C_i^k\gamma(f_l)^2\right]\\ =&Z_i(h, y_{-i}) - Z_i(l, y_{-i})\end{aligned} \tag{21}$$

Therefore, our COG is an *exact potential game*. ■

Theorem 2. COG $\mathcal{G}\langle$**U, Y, Z**$\rangle$ has a pure-strategy Nash equilibrium.

Proof: It has been proved that every potential game has a pure-strategy (deterministic) Nash equilibrium and it possesses the *finite improvement property* [15], [17]. Meaning that every unilateral *better response update* is finite and will always terminate at a deterministic *Nash equilibrium*. A *better response* of a player denotes to a strategy which

reduces player's cost compared to the current strategy. Since COG is a type of potential games, consequently it has a *(pure-strategy) Nash equilibrium*. ■

Next, we describe our Strategic Offloading Algorithm to obtain a deterministic Nash equilibrium solution.

V. Strategic Offloading Algorithm

In this section, we propose our Strategic Offloading Algorithm (SOA) which allows each UAV to improve their own offloading decisions such that all UAVs achieve a mutually satisfactory result (i.e., *Nash equilibrium*), where no UAV has incentive to change its strategy to other strategies to reduce its cost. SOA runs iteratively in order to find the final solution, and in any iteration it allows each UAV decides a *better response update*. We now define the concept of *better response update* in SOA:

Definition 4. An offloading strategy change of UAV u_i from y_i to y_i' is a *better response update* if and only if its corresponding cost function decreases, i.e.,

$$Z_i(y_i', y_{-i}) < Z_i(y_i, y_{-i}). \tag{22}$$

Then, y_i' is called a *better strategy* of u_i compared to y_i. According to this property, we formulate the update rule of SOA, where only one better strategy is arbitrarily selected as the final updating decision at each iteration of the algorithm. In addition, we define the *best response* of a UAV as the strategy among the set of better strategies which minimizes that UAV's energy cost Z.

Lemma 1. Given the strategy profile $\mathbf{Y}_{-i}$ of all other UAVs except u_i in COG, exploiting the *better-response update* will always lead to at least an equal or better result in terms of the total system energy cost, and the *better-response update* of u_i can be computed by:

$$y_i^{UD} = \begin{cases} q, & \text{if } y_i = p \text{ and } E^k_{i\to p} > E^k_{i\to q}, \\ \text{no update}, & \text{otherwise.} \end{cases}$$

Proof: Let $\boldsymbol{\mathcal{S}}(t) = (\mathcal{S}_1(t), \ldots, \mathcal{S}_n(t))$ denote the multiset of all *better-response strategies* $\mathcal{S}_i$ (for all $u_i \in U$) at iteration t, where every element in $\mathcal{S}_i(t)$ has a lower energy cost than the previous energy cost at $t-1$. To be more precise, the *better-response strategy* means that if a task of u_i (i.e., k) is currently decided to be offloaded to u_p but there exists any UAV u_q such that $E^k_{i\to p} > E^k_{i\to q}$, then changing the offloading strategy of u_i from $y_i = p$ to $y_i = q$ while satisfying the desired constraints leads to a reduced cost (when other UAVs keep their strategies unchanged at this time). As a result, $y_i = q$ is a *better-response solution* for u_i at this iteration and we save it into the subset $\mathcal{S}_i(t)$ of the multiset $\boldsymbol{\mathcal{S}}(t)$. When all *better-response solutions* of each UAV are computed, one is chosen as the only update for the next iteration. Otherwise, u_i will adhere to its current decision at the next iteration. By implementing this update rule, the total system energy cost always remains unchanged or is reduced. ■

Algorithm 1 Strategic Offloading Algorithm (SOA)

Input: $\mathcal{G}\langle$**U, Y, Z**$\rangle$, $\mathbf{T}_{max}$, $\mathbf{u}^{cap}$

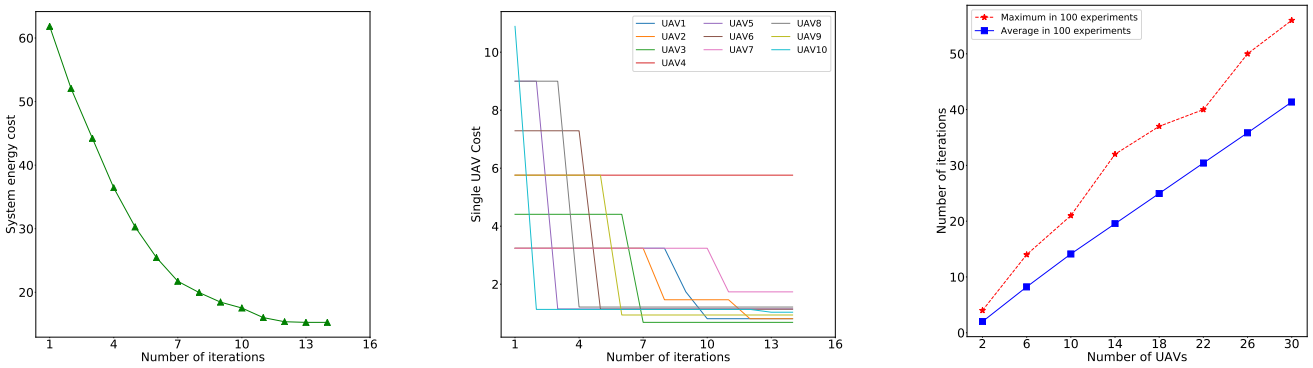
1: $t = 0$
2: $y_i(0) = i, \forall u_i \in U$ $\quad$ {u_i to execute its task locally}
3: $d_i(0) = 0, \forall u_i \in U$ {none of UAVs are offloading to u_i}
4: **repeat**
5: $\quad$ $\mathbf{Y}(t) \leftarrow$ current decision profile
6: $\quad$ $\boldsymbol{\mathcal{S}}(t) \leftarrow \emptyset$
7: $\quad$ **for** each UAV u_i **do**
8: $\quad\quad$ Calculate $d_i(t), d_i(t+1)$ {current & next capacity}
9: $\quad\quad$ Calculate $T_i(t), T_i(t+1)$ {current & next exec. time}
10: $\quad\quad$ Calculate $Z_i^k(t), Z_i^k(t+1)$ {current & next energy cost}
11: $\quad\quad$ $\mathcal{S}_i(t) \leftarrow$**filter** all *better-response strategies* of u_i
12: $\quad\quad$ $\boldsymbol{\mathcal{S}}(t) \leftarrow \boldsymbol{\mathcal{S}}(t) \cup \mathcal{S}_i(t)$
13: $\quad$ **select** only one *better response strategy* from $\boldsymbol{\mathcal{S}}$(t) for the next decision profile: $y_h^{UD}(t+1) = y_h(t)$
14: $\quad$ $t = t + 1$
15: **until** $\boldsymbol{\mathcal{S}}(t) = \emptyset$
16: $\tau \leftarrow t$

Output: $\mathbf{Y}^* = \mathbf{Y}(\tau) \implies$ *a Nash equilibrium*

The offloading decision profile of UAVs becomes stable when a *Nash equilibrium* is achieved. We propose SOA in virtue of the *better response update* (according to Lemma 1) to reach a *Nash equilibrium*. SOA is given in Algorithm 1. SOA sets the initial strategies of the UAVs to local computing (Line 2). It then calculates the better-response strategies of the UAVs by finding the decision profile and the energy consumption of all possible strategies of each UAV (Lines 7-12). The set $\mathcal{S}_i(t)$ contains the better-response strategies for UAV u_i (Line 11). Having a non-empty set $\boldsymbol{\mathcal{S}}(t)$, SOA selects one of them as the only update for the next iteration (according to the Lemma 1). Iteratively, SOA converges to a stable decision profile at which all UAVs are at their own best offloading decisions. This happens after a finite number of iterations τ due to holding the *finite improvement property*.

Theorem 3. SOA always converge to a stable profile $\mathbf{Y}^* = \{y_1^*, \ldots, y_n^*\}$, at which each strategy y_i^* is the *best response* of UAV u_i, and this profile is a *Nash equilibrium* of COG.

Proof: We prove by contradiction. Assuming SOA terminates after a finite number of iterations τ and at this terminus $\mathbf{Y}(\tau)$ is not a *Nash equilibrium*, then there exists a UAV u_i who can deviate from its current strategy $y_i(\tau)$ to a new strategy $y_i'(\tau+1)$ to reduce its energy cost. According to the *better response update* property of Lemma 1, SOA will continue to allow this change y_i to y_i' to improve the energy cost. This contradicts the initial assumption and thus



(a) Convergence of COG (n=10) (b) UAV's individual cost (n=10) (c) Scalability of SOA

Figure 1: Performance Evaluation of SOA

$y_i(\tau) = y_i^*(\tau)$. ■

VI. Experimental Results

Experimental Setup. We have a set of heterogeneous UAVs. For each UAV u_i, its computation capability F_i is chosen arbitrarily from $[3.0, 6.0]$ GHz (0.5 incerements) and its task capacity u_i^{cap} is randomly selected from $[1, 10]$. Tasks can have different computation requirements. Here, we consider the face recognition application in [18], where the computation task k has $(C^k, S^k) = (3000, 420)$ with the units of Megacycles and KB, respectively. Moreover, we set the maximum tolerable execution time of each task T_{ik}^{max} to obey the uniform distribution $U[2, 10]$ (seconds). As for the wireless access, the transmission power P_i is randomly selected from $[0.05, 0.5]$ Watts (0.05 increments), and the transmission bandwidth B_i obeys Gaussian distribution with mean $\mu_i = 5$ MHz and standard deviation $\sigma_i = 0.15\mu_i$. Each scenario has been repeated 100 times, and the average results are reported.

Analysis of Results. In our experimental setup, we set that each UAV initially chooses to execute its task locally, i.e., $x_{ii} = 1,\ i \in \{1, \ldots, n\}$. To enable a tractable analysis, we always select the *best response update* from the *better-response solutions* as the only update at the next decision making iteration in our proposed SOA.

We first show the performance of SOA in terms of convergence of the potential function in Fig. 1a. This figure presents that after a finite number of iterations, our proposed SOA always converges to a stable point (decision profile) at which the value of the potential function is minimized. The minimum point is a *Nash equilibrium* of our game.

From the perspective of individual users, Fig. 1b shows that SOA ensures UAV's individual energy cost is reduced or remains unchanged, and shows it converges to a Nash equilibrium. Note that, some UAVs might not be able to reduce their energy cost but keep it unchanged (e.g., UAV 4).

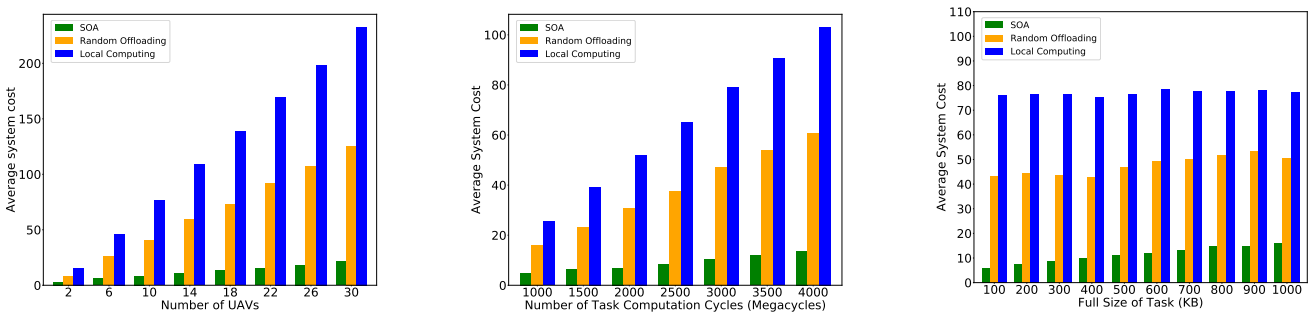
To study the scalability of SOA, we verify the performance of SOA with different number of UAVs, and show the average and maximum number of iterations for the convergence to a *Nash equilibrium*. As shown in Fig. 1c, the average number of iterations increases almost linearly as the number of UAVs increases, which shows that SOA scales well with the number of UAVs.

We perform sensitivity analysis on the system cost with respect to three parameters. To evaluate the system cost based on different number of UAVs, we compare the results of SOA with those of two other approaches: Local Computing Approach and Randomly Offloading Approach, where the former indicates all UAVs execute their tasks locally and the latter selects a UAV randomly for offloading a task. Fig. 2a shows that SOA outperforms Random Offloading and Local Computing approaches in terms of the system cost. The reason is that each UAV has more offloading choices, and SOA is able to find the minimum system cost by updating a *best response* strategy at each decision making iteration.

We then analyze the performance of our proposed SOA in terms of the overall system energy with respect to computation cycles of the tasks. As shown in Fig. 2b, SOA outperforms the Local Computing Approach. The reason is that as the task computation cycles increase, more UAVs prefer to offload their tasks to other UAVs in order to reduce the energy consumption.

We also evaluate the performance of SOA with respect to different task sizes (i.e., S^k). Fig. 2c shows that the average system cost obtained by SOA is significantly less than that of Local Computing Approach as the data size of tasks increases. Owing to the property of potential games and Eq. (13), as the task data size increases, the total transmission energy consumption for offloading via the wireless access increases. Thus, more UAVs choose to execute their tasks locally to prevent a higher energy consumption caused by offloading. Moreover, the average system cost obtained by Local Computing Approach remains almost invariable, as expected.

From the above results, we conclude that our proposed



(a) on number of UAVs (b) on task computation cycles C^k (n=10) (c) on task sizes S^k (n=10)

Figure 2: Sensitivity Analysis of Average System Cost

SOA is able to find stable solutions by converging to Nash equilibria achieving satisfactory energy cost while ensuring QoS for the IoT users.

VII. Conclusion

In this paper, a swarm of capacitated UAV-mounted cloudlets provides computing services to IoT users. UAVs' goal is to reduce their energy consumption of running tasks satisfying their QoS. We formulated the problem as an optimal integer programming model, and proposed an effective capacitated offloading game (COG) to solve this problem. We proved that COG has a *Nash equilibrium* and designed a strategic offloading algorithm (SOA) to optimize the total system cost and find a Nash equilibrium. Our experimental analysis showed that SOA is scalable and converges to a Nash equilibrium. In future work, we plan to consider the impact of UAVs mobility on offloading decision makings.

Acknowledgment

This research was supported in part by NSF grant CNS-1755913.

References

[1] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[2] H. Shakhatreh and A. Khreishah, "Optimal placement of a UAV to maximize the lifetime of wireless devices," in *Proc. of the 14th IEEE Intl. Conf. on Wireless Communications Mobile Computing*, 2018, pp. 1225–1230.

[3] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. on Vehicular Technology*, vol. 67, no. 3, pp. 2049–2063, 2018.

[4] M.-A. Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci, "Computation offloading game for an UAV network in mobile edge computing," in *Proc. of IEEE Intl. Conf. on Communications*, 2017, pp. 1–6.

[5] W. Labidi, M. Sarkiss, and M. Kamoun, "Joint multi-user resource scheduling and computation offloading in small cell networks," in *Proc. of the 11th IEEE Intl. Conf. on Wireless and Mobile Computing, Networking and Communications*, 2015, pp. 794–801.

[6] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.

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

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

[9] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proc. of the 18th ACM Intl. Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2015, pp. 271–278.

[10] N. Sharghivand, F. Derakhshan, and L. Mashayekhy, "QoS-aware matching of edge computing services to internet of things," in *Proc. of the 37th IEEE International Performance Computing and Communications Conference*, 2018, pp. 1–8.

[11] L. Mashayekhy, M. Nejad, and D. Grosu, "A trust-aware mechanism for cloud federation formation," *IEEE Transactions on Cloud Computing*, pp. 1–14, 2019, in press.

[12] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Wireless communication using unmanned aerial vehicles (UAVs): Optimal transport theory for hover time optimization," *IEEE Trans. on Wireless Communications*, vol. 16, no. 12, pp. 8052–8066, 2017.

[13] H. He, S. Zhang, Y. Zeng, and R. Zhang, "Joint altitude and beamwidth optimization for UAV-enabled multiuser communications," *IEEE Communications Letters*, vol. 22, no. 2, pp. 344–347, 2018.

[14] G. Iosifidis, L. Gao, J. Huang, and L. Tassiulas, "A double-auction mechanism for mobile data-offloading markets," *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1634–1647, 2015.

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

[16] *Algorithmic Game Theory*. Cambridge Univ. Press, 2007.

[17] R. W. Rosenthal, "A class of games possessing pure-strategy nash equilibria," *Intl. Journal of Game Theory*, vol. 2, no. 1, pp. 65–67, 1973.

[18] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. of IEEE symposium on Computers and communications*, 2012, pp. 59–66.