A Real-Time and Non-Cooperative Task Allocation Framework for Social Sensing Applications in Edge Computing Systems

Daniel (Yue) Zhang, Yue Ma, Yang Zhang, Suwen Lin, X. Sharon Hu, Dong Wang
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN, USA
{yzhang40, yma1, yzhang42, slin4, shu, dwang5}@nd.edu

***Abstract*—Social sensing has emerged as a new sensing application paradigm where measurements about the physical world are collected from humans or devices on their behalf. A key limitation in the current social sensing solution space is that data processing and analytics are often done in a *"backend"* mode (e.g., on dedicated servers or commercial cloud platforms). Such mode ignores the rich processing capability of increasingly powerful edge devices (e.g., mobile phones and nodes in Internet of Things). Exploiting such edge devices in the social sensing setting introduces new challenges to real-time resource management. In this work, we develop a Bottom-up Game-theoretic Task Allocation (BGTA) framework to solve the critical problem of allocating real-time social sensing tasks to self-aware and non-cooperative edge computing nodes. In particular, we address two important challenges in solving this problem. The first one is *"conflicting interest"* where the objectives of applications and edge nodes may be at odds with each other. The second challenge is *"asymmetric and incomplete information"* where the application is often unaware of the detailed status (e.g., energy profile, utilization, CPU frequency) and compliance level of the edge nodes. To address these challenges, we first design a non-cooperative task allocation game model to address the conflicting objectives of the applications and edge nodes. We then develop a decentralized Fictitious Play scheme to allow each edge node to make its own decision on which task to execute in a non-cooperative context. Finally, we design a dynamic incentive mechanism to ensure the decisions made by the edge nodes meet objectives of the application. We implement a system prototype deployed on Nvidia Jetson TX1 and Jetson TK1 boards and evaluate our task allocation framework using two real world social sensing applications. The results show that our scheme can well satisfy Quality of Service (QoS) requirement of the applications while providing optimized payoffs to edge nodes compared to the state-of-the-art baselines.**

I. Introduction

This paper considers the development of a real-time and non-cooperative task allocation framework for social sensing applications in edge computing systems. Motivated by the ubiquitous Internet connectivity and the proliferation of portable devices, social sensing has become a new sensing paradigm of collecting real-time measurements about the physical world from humans or mobile devices on their behalf. Examples of social sensing applications include obtaining real-time situation awareness in the aftermath of a disaster using online social media [1], monitoring the air quality of a city using inputs from common citizens [2], and locating street potholes using mobile phone apps of drivers [3]. A key limitation in the current social sensing solution space is that data processing and analytics are often done in a *"backend"* mode (e.g., on dedicated servers or commercial cloud platforms). Such mode ignores the rich processing capability of increasingly powerful edge devices (e.g., mobile phones and nodes in Internet of Things).

The advent of edge computing pushes the frontier of computation, service, and data to the edge of the network [4], [5] and brings new opportunities for real-time social sensing applications (Figure 1). The benefits of running social sensing applications in edge computing systems are three folds: (i) the collected data can be processed at the edge nodes, which saves the bandwidth of sending all data to the backend servers; (ii) the edge nodes can execute the computational tasks that are typically performed by the backend servers and makes the social sensing applications more scalable and responsive; (iii) the massive computation power at the edge nodes are better utilized and the end users could also obtain payoff and rewards by undertaking social sensing tasks. However, exploiting the edge devices in the social sensing setting introduces new challenges to real-time resource management.

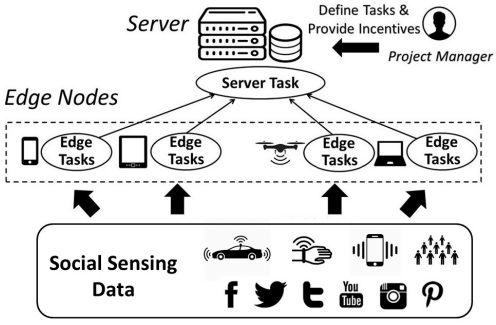

Figure 1. Social Sensing in Edge Computing

In this paper, we focus on a critical problem in running a social sensing application on edge devices, i.e., allocating real-time social sensing tasks to non-cooperative edge computing

nodes. We assume edge nodes (often owned by end users) are in general non-cooperative and selfish (e.g., they are not interested in executing the sensing tasks or sharing their private device status unless incentives/payoffs are provided) [6]. This assumption is unique in the edge computing system and in sharp contrast to the assumption made in the "backend" based solutions where all computational nodes are fully cooperative and information is shared among all nodes [7], [8]. While many works have been proposed to address the task allocation problem in real-time systems [7], [9]–[11], two major challenges that are unique to running social sensing applications in the edge computing systems have not been well addressed: *conflicting interests* and *asymmetric and incomplete information*.

Conflicting interests: The first challenge is conflicting interests where the application and edge users have different and potentially conflicting objectives. For example, from the application's perspective, it is important to ensure the edge nodes finish the allocated social sensing tasks within a certain period of time (e.g., deadline) to meet the QoS requirements. Current social sensing task allocation schemes are often "top-down" based by assuming that a centralized decision maker (e.g., project manager or a server level allocation algorithm) makes the decision on which node should take on which tasks and when the tasks should be executed [12], [13]. However, these "top-down" based methods do not fit well with the edge computing paradigm where edge nodes can be personal mobile devices owned by end users who are not necessarily willing to execute the tasks allocated on their devices [14]. For example, a smartphone user may refuse to execute tasks which would possibly affect the experience of using the phone.

Asymmetric and incomplete information: The second challenge is the asymmetric and incomplete information where the application (server) and edge nodes usually have different degrees of information in the edge computing paradigm. The server normally has detailed information about the tasks (e.g., the dependency and criticality of the tasks) but knows little about the edge nodes. In contrast, edge nodes are often less concerned about the details of the tasks and more concerned about their own device status (e.g., node's current utilization, energy consumption, memory usage). Furthermore, an edge node may not share its status information with the server or other edge nodes in the system due to various concerns (e.g., privacy, energy, bandwidth) [15]. Such asymmetric and incomplete information at both server and edge nodes make the task allocation problem of social sensing applications in edge computing systems more challenging.

To address these challenges, we introduce a novel Bottom-up Game-theoretic Task Allocation framework (BGTA) framework. Our framework reconciles the conflicting objectives of meeting the QoS requirements from applications and maximizing the payoffs for the edge nodes via a game theoretic approach. We specifically make the following contributions. *First*, we develop a non-cooperative task allocation game model to address the conflicting objectives demanded by the applications and edge nodes. This game model ensures that selfish edge nodes can achieve mutually satisfactory decisions on the task allocation by reaching a Nash Equilibrium. *Second*, we devise a decentralized Fictitious Play scheme that allows each edge node to make its own decision on which task to take in a non-cooperative context. This scheme does not require the sharing of private node status information with the server, thus overcomes the asymmetric and incomplete information challenge and provides privacy preservation at the edge nodes at the same time. *Third*, we design a dynamic incentive mechanism to ensure the decisions made by the edge nodes meet objectives of the application. *Finally*, we implement a system prototype of BGTA using Nvidia Jetson TX1 and Jetson TK1 boards and evaluate it using two real world social sensing applications: Truth Discovery and Spatial-temporal Inference.

We compared our scheme with the state-of-the-art task allocation schemes used in edge computing environments. The results show that our scheme achieves significant performance gain in terms of meeting the objectives of both applications and edge nodes (e.g., our scheme achieved more than 12% improvements in deadline hit rate for the application and 20% more payoffs for edge nodes compared to the baselines.).

II. Related Work

A. Social Sensing

Social sensing has received a significant amount of attention due to the proliferation of low-cost mobile sensors and the ubiquitous Internet connectivity [16]. Examples of social sensing applications include intelligent transportation systems [17], urban sensing [18], and disaster and emergency response [12]. This work uses two important social sensing applications as our case studies: *Truth Discovery* and *Spatial-temporal Inference*. The goal of the truth discovery application is to jointly identify truthful information and reliable data sources by analyzing an influx of noisy social sensing data [12], [19]. Spatial-temporal inference is a social sensing application that infers missing sensor measurements from the incomplete social sensing data using spatial-temporal predictive models (e.g., predicting hazardous environmental conditions [18]). The BGTA framework complements current social sensing solutions deployed in the "backend" of the networks and efficiently utilizes the rich computing capability available at the edge nodes.

B. Edge Computing and Computation Offloading

A comprehensive survey of edge computing is given by Ahmed *et al.* [4]. A critical challenge in edge computing is *computational offloading* where heavy data analytics tasks are transferred to external servers or devices from sensors with limited memory, battery and computation power [20]. However, pushing all the computation tasks to the remote servers may be impractical due to the limited network bandwidth and large communication latency. Various works have been focused on offloading computation tasks to the edge devices to reduce communication costs and application latency [21]. For example, Gao *et al.* proposed a probabilistic computational

offloading framework that offloads the computation tasks to the mobile devices [22]. Kosta *et al.* proposed an energy-aware code offloading framework to dynamically switch between edge nodes and cloud servers to improve the energy efficiency of the system [23]. Liu *et al.* proposed a decentralized data offloading scheme using the multi-item auction and congestion game approach to achieve efficient offloading performance [24]. The key limitation of the above works is that they fail to consider the non-cooperativeness and compliance issues of edge nodes which is addressed in our BGTA framework.

C. Task Allocation in Real-Time Systems

Task allocation is an important problem in real-time systems and both centralized and distributed solutions have been developed to address this problem [7], [9], [25]. For example, Zhu *et al.* proposed a Mixed Integer Linear Programming based approach to meet the deadlines and minimize the end-to-end latencies in hard real-time systems [7]. Su *et. al* developed a mixed criticality task model to maximize the number of low-criticality tasks being executed without influencing the timeliness of high-criticality tasks [26]. A set of task allocation schemes have been developed to optimize the hardware reliability and energy efficiency in real time systems [25], [27]. Most of the above schemes adopt a centralized approach that depends on a central decision maker to allocate tasks in the system. Such approach will fail in the edge computing systems where the nodes might refrain from providing necessary information to accomplish the centralized task allocation [11].

Decentralized task allocation schemes have been developed to address the above limitation. For example, Walsh *et al.* proposed a classical market protocol for allocating tasks among agents who trade tasks and resources at prices determined by an auction protocol [6]. Ahmad *et al.* proposed a game theoretic approach for scheduling tasks on multi-core processors to jointly optimize performance and energy [11]. Jin *et al.* proposed a decentralized task allocation protocol in a dynamic and uncertain environment [9]. However, these decentralized schemes cannot be directly applied to our task allocation problem because they assume cooperative and controlled resources and ignore the non-cooperative nature and the compliance issues of end users in edge computing. In contrast, our work proposes a bottom-up game theoretic approach that decides the task allocation locally by each edge node. Our approach not only yields optimal payoffs for the edge nodes but also meets the QoS requirement of applications.

III. Problem Formulation

In this section, we present the real-time and non-cooperative task allocation problem. We first present the systems, models, and assumptions for our problem formulation. We then formally define the objectives of our problem.

A. Systems, Models and Assumptions

Figure 1 (in Section I) depicts a high-level overview of the BGTA framework. In this framework, a project manager launches a real-time social sensing application that constantly collects sensor data about the physical world via mobile devices (e.g., laptops, drones, smartphones, automobiles). We refer to these devices as *edge nodes*. Let $EN = \{E_1, E_2 ... E_X\}$ denote the set of all edge nodes in the application. These edge nodes are not only able to collect sensor data but also perform some computation tasks (indicated as Edge Tasks in Figure 1) to reduce the burden of backend servers. The results of the edge tasks are often sent back to the server where they get aggregated for further analysis. Both the server task and edge tasks are defined by the project manager who has full control over the backend server and can also provide incentives to motivate edge nodes to finish the assigned tasks. A few key terms related to the BGTA framework are defined as follows.

DEFINITION 1. Edge Node: An edge node is a computational resource that can collect and processes data near the logical extreme of a network. The edge nodes may not constantly connect to the network and often have limited battery, bandwidth, memory and computation power. Examples of edge nodes include portable embedded systems, laptops, smartphones, tablets and sensors.

DEFINITION 2. Server: A server is a backend processing unit for task distribution and data processing/analytics. A server is often a powerful and controlled system (e.g., dedicated virtual machines, cloud platforms) that is directly managed by the project manager. In the rest of this paper, we use term "server" to refer to both physical server and the project manager/application and use the term "edge node" to refer to both an edge device and its owner (i.e., end user). The meaning of the term should be clear in the specific context.

DEFINITION 3. Edge Node Status: An edge node status defines the operating status of the edge node at any given time. It consists of an edge node's energy profile, CPU frequency, CPU utilization, and available memory.

DEFINITION 4. Edge Task and **Server Task**: An edge task is performed on an edge node that is close to the data sources. Examples of the edge tasks include data preprocessing, encryption, and feature extractions. A server task runs on the server to aggregate the results of edge tasks and perform computationally intensive tasks (e.g., data analytics). The BGTA needs to handle both server task and edge tasks .

We further introduce two important models (i.e., task model and energy model) adopted by the BGTA framework.

1) Task Model: We assume a periodic mixed criticality task model where a server task TK_s^m and a set of N edge tasks are initialized by the server at the beginning of each period m of length Δ, $1 \leq m \leq M$: $TK_e^m = \{\tau_1^m, \tau_2^m, ...\tau_{N_m}^m\}$. M is the total number of periods for the real-time social sensing application and N_m is the total number of edge tasks in period m. An edge task is described by a 3-tuple: $\tau_i^m = \{v_i^m, c_{i,x}^m, p_i^m\}$, where v_i^m is the data volume to be processed by task τ_i^m; $c_{i,x}^m$ is the estimated worst-case execution time (WCET) if τ_i^m is assigned to the edge node $E_x, 1 \leq x \leq X$. We assume the deadline equals to the predefined period Δ where each task

should be processed before the next period starts, which is a common assumption in real-time social sensing applications [13]. We also assume that the task that cannot finish before the deadline will not be executed in the next period. The task criticality p_i^m defines the importance of the task τ_i^m to the application, which is formally defined below:

DEFINITION 5. Task Criticality: a score that measures the data quality loss of a social sensing application should a task fails to meet the deadline.

2) Energy Model: We adopt the following well-established model to measure energy consumption of edge nodes. Consider a computing system that consists of a CPU with a variable operating frequency f, we first model the power consumption of the system based on [28] as:

$$Power(f) = P_{static} + af^b, b \geq 2, f_{min} \leq f \leq f_{max} \quad (1)$$

where $Power(f)$ represents the power consumption for frequency f. P_{static} is the static power consumption. f_{min} and f_{max} are the minimum and maximum CPU frequency of an edge node and $\{a, b\}$ are constants. In this model, we assume the WCET of a task is proportional to the data size and inversely proportional to the operating frequency [27], i.e.,

$$c_{i,x}^m \propto \frac{v_i^m}{f_x^m}, 1 \leq m \leq M, 1 \leq i \leq N_m, 1 \leq x \leq X \quad (2)$$

where f_x^m denotes the frequency of E_x at period m. Therefore, the worst-case energy consumption of a task τ_i^m (starting at time m_{start}) can be calculated as:

$$e_x^m(f) = \int_{m_{start}}^{m_{start}+c_{i,x}^m} Power_{f_x^m} * dt \quad (3)$$

Finally, we clarify the assumptions we made in BGTA:

- *Heterogeneity:* We assume that the edge nodes are heterogeneous in terms of architecture, energy profile, and processing capabilities.
- *Non-cooperative and Selfish Edge Nodes:* We assume that edge nodes only care about their own welfare (e.g., minimizing energy cost, CPU usage).
- *Asymmetric and Incomplete Information:* We assume that each edge node has complete information of its own status but may refuse to share such information with the server or other nodes. We assume the server has the complete information about the tasks and understands how such information would affect the QoS requirement of the application. However, the edge nodes are less concerned or completely unaware of the detailed task information and its implication to the QoS.
- *Dynamic Edge Node Status:* We assume the end users may constantly use other applications on their edge devices, resulting in a dynamic edge node status.

B. Objectives

Given the definitions, assumptions and models, we formally define the objectives of this paper. We assume server and edge nodes have potentially conflicting interests where the server tries to optimize the QoS and the edge nodes are interested in maximizing their payoffs. Therefore, our goal is to develop a real-time and non-cooperative task allocation scheme that can best satisfy both sides. This task allocation problem can be formulated as a multi-objective optimization problem that targets at finding a task allocation scheme S to:

$$\text{maximize:} \quad \sum_{m=1}^{M} u_x^m, \forall 1 \leq x \leq X \quad \text{(edge node's objective)}$$

$$\text{minimize:} \quad \mathcal{L} = \sum_{m=1}^{M} \sum_{i=1}^{N_m} p_i^m * \delta_i^m \quad \text{(server's objective)}$$

$$\text{s.t.:} \quad \text{finish time of } \tau_i^m \leq \Delta, \forall \tau_i^m \in TK_e^m \quad \text{(constraints)}$$

where u_x^m is the payoff that edge node E_x receives by executing edge tasks. u_x^m is a function of e_x^m defined in Equation (3) and more details on u_x^m will be given in Section IV. $\mathcal{L}$ is a loss function to measure the QoS degradation due to deadline misses. Specifically, δ_i^m is a binary variable that denotes whether task τ_i^m misses the deadline ($\delta_i^m = 1$) or not ($\delta_i^m = 0$). The intuition is that the more critical tasks miss their deadlines, the more likely the QoS of a social sensing application will degrade. It is well known that real-time task allocation problem on the heterogeneous distributed system is in general NP-hard [29], [30]. The problem becomes more challenging in this multi-objective formulation when the objectives of server and edge nodes are potentially conflicting. In the next section, we develop a novel Bottom-up Game-theoretic Task Allocation scheme to solve this problem.

IV. BGTA Framework

An overview of BGTA is given in Figure 2. The BGTA scheme consists of three major components: i) a Non-cooperative Task Allocation Game (NTAG) model to address the conflicting interests between server and edge nodes and lay out the payoffs of different task allocation strategies; ii) a Decentralized Fictitious Play (DFP) algorithm that runs on each edge node to efficiently learn the optimal task allocation strategy by achieving the Nash Equilibrium; and iii) a Dynamic Incentive Mechanism (DIM) to ensure that the QoS requirement of the social sensing application is met on the server side. We explain these components in detail in this section.

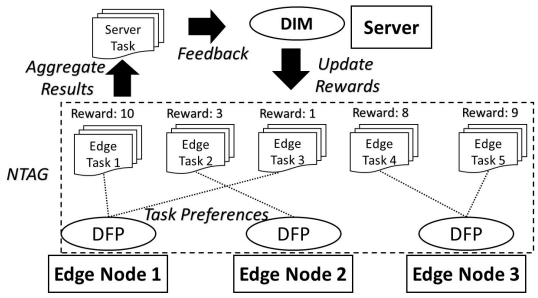

Figure 2. Overview of BGTA

A. A Non-cooperative Task Allocation Game (NTAG)

The conflicting interests of applications and edge nodes can be modeled as a Non-cooperative Task Allocation Game

(NTAG). In particular, a NTAG is described by a tuple $(H, EN, TK_e, S, \Psi, \pi, R)$ where H is the game host (i.e., the server) and EN is a set of X players (i.e. edge nodes) in the game. TK_e is a set of N edge tasks and Ψ stands for the task allocation strategy space for all players: $\Psi = \Psi_1 \times \Psi_2 \times ... \times \Psi_X$ where Ψ_x represents the strategy space of player E_x (i.e., 2^N possible task allocations). π represents a node-specific cost function vector $\pi = \{\pi_1, \pi_2, .., \pi_X\}$ for each edge node where the element π_x represents the cost of the edge node E_x to execute the allocated tasks based on strategy S_x. Such cost often depends on each individual edge node's energy profile and the workload of the task. $R = \{R_1, R_2..., R_N\}$ is a set of rewards that is provided by the application where R_i represents the reward for τ_i. Such rewards can be monetary rewards (e.g., cash or virtual currency) or non-monetary (e.g., contribution/experience scores, competitive rankings). In each period, we define a Strategy Profile S as a set of individual task allocation strategies, i.e. $S = \{s_1, s_2, ..., s_X\} \in \Psi$, where s_x denotes the strategy (i.e. which tasks to execute) on the edge node E_x. We formally define a Task Allocation Strategy and a Strategy Profile as follows.

DEFINITION 6. Task Allocation Strategy: a task allocation strategy defines an edge node's choice of tasks in TK_e.

DEFINITION 7. Strategy Profile: a strategy profile captures the set of task allocation strategies employed at all edge nodes.

For example, consider a NTAG of 2 edge nodes and a task set of 4 edge tasks in a given period. A task allocation strategy for edge node 1 can be "1100" denoting this edge node is willing to pick first two tasks. Similarly, the task allocation strategy of edge node 2 can be "0110", which means node 2 is willing to pick the second and third task. The Strategy Profile S is {"1100", "0110"} in this example. We assume each task allocation strategy is a Pure Strategy, i.e., an edge node will deterministically "pick" or "not pick" a task rather choosing it with a probability (i.e., a Mixed Strategy) [31].

The original goal of each edge node is to find the optimal strategy that minimizes its cost. This will result in no tasks being picked (since this strategy gives zero cost at each edge node). However, such task allocation results obviously conflict with the objective of the application. To address the above conflict of interests challenge, the NTAG model provides an incentive mechanism to encourage edge nodes to pick edge tasks by assigning rewards for tasks, namely R.

After defining the reward and cost of NTAG, we further define the set of payoff functions of a strategy profile S as $U(S) = \{u_1, u_2, ..., u_X\}$ where u_x denotes an individual payoff function of edge node E_x, and

$$u_x = g(R, \pi_x, s_x, s_{-x}). \tag{4}$$

Here s_{-x} represents the task allocation strategies for other edge nodes. The payoff function represents how much benefits the edge node E_x can gain if strategy s_x is taken by the node.

To derive the payoff function, we first introduce our game protocol of NTAG at each period: (i) the server initializes a set of social sensing tasks and decides the reward for each task; (ii) each edge node simultaneously picks a task that has the best payoff for itself; (iii) if multiple nodes choose the same task, we randomly assign this task to one of the competing nodes. We refer to one round of this task allocation process as an "iteration"; (iv) within each iteration, we assume each edge node is myopic and only picks one task at a time; (v) keep iterating until all tasks are picked. Then each node starts to process the tasks they picked. The execution sequence depends on the order of which the tasks are picked by the edge nodes.

The above protocol follows the rule of *singleton weighted congestion game* [32] where the expected reward of each task monotonically decreases as the number of edge nodes that picked the task increases, and each player has its own cost function (weighted congestion property). Each player can only pick one item (i.e. task) at a time (singleton property). The reasons for using such a rule are two folds: (i) the introduction of the singleton property reduces the strategy space from $O(2^N)$ to $O(N)$; (ii) the Pure Strategy Nash Equilibrium is guaranteed to exist under the singleton weighted congestion game protocol [31], which is crucial for the edge nodes to make mutually satisfactory task allocation decisions.

Let us assume node E_x picks the i^{th} task in an iteration and $d(i)$ is the number of nodes who pick the i^{th} task. We can define the individual payoff function of edge node E_x based on the above protocol as:

$$u_{x,i} = \begin{cases} \dfrac{Exp(R_i)}{\pi_x} = \dfrac{R_i}{d(i) * e_x}, & \dfrac{c_i^x}{\Delta} + BU_x \leq 1 \\ 0, & \dfrac{c_i^x}{\Delta} + BU_x > 1 \end{cases} \tag{5}$$

Here $u_{x,i}$ is the payoff function of E_x if it picks edge task τ_i in the current iteration. $Exp(R_i)$ is the expected reward of the i^{th} task and $\pi_x = e_x$ is the individual energy cost of edge node E_x. c_i^x is the worst case execution time of the task defined in Section III. BU_x captures the total utilization by all the tasks currently assigned to E_x and e_x is a damping factor to the payoff function: the more energy an edge task consumes, the less payoff the node will gain. We assume that earliest deadline first (EDF) scheduling algorithm is used at E_x. If an edge node picks a task it could not finish before the deadline (i.e., $\frac{c_i^x}{\Delta} + BU_x > 1$), the edge node will not receive any reward. Therefore, the objective of edge node E_x is to pick the set of tasks that maximize its own payoff function as follows:

$$\underset{i}{\text{argmax}} \ u_{x,i} \tag{6}$$

To ensure that each edge node makes its best decision towards the above objective, our goal is to find a Nash Equilibrium for the NTAG. The Nash Equilibrium exists in a non-cooperative game where each player (i.e., edge node in our model) is assumed to know the equilibrium strategies of the other players and no player has anything to gain by only changing his/her own strategy [33]. In the next subsection, we describe how to find the Nash Equilibrium in NTAG.

B. Decentralized Fictitious Play (DFP)

A few classical approaches exist to find the Nash Equilibrium. Example of such solutions include Best Response [24], [34], Fictitious Play (FP) [35] and Reinforcement Learning [36]. A key challenge in finding the Nash Equilibrium in our NTAG model is the incomplete information problem (i.e., an edge node often has incomplete or no information on the individual payoff function of other nodes). This results in a scenario where an edge node cannot precisely derive other nodes' best strategies to make its own best response (similar as an auction scenario where each bidder would not share her own valuation of an item beforehand). The Fictitious Play scheme is designed to address this challenge. The key idea of FP is that each player is intended to "guess" the strategies other players might pick based on their historical strategies (without knowing their actual payoff functions) and make its own decision to yield the highest payoff for itself.

In particular, each edge node keeps a local "belief table" L where the element $l_{x,i}$ denotes the probability that the edge node E_x picks task τ_i. At the beginning of the game, we initialize L as:

$$l_{x,i} = \frac{R_i}{\sum_{i=1}^{N} R_i} \tag{7}$$

Each edge node first assumes other nodes will always try to pick the tasks with higher payoffs. Each node then picks the task that will potentially yield the best payoff based on its belief of what other nodes might pick. After observing the decisions made by other edge nodes, each node will update its belief table accordingly. Traditional FP algorithm assumes a centralized program that has the payoff information of all nodes in the system, which does not work in the edge computing scenario. We propose a Decentralized Fictitious Play (DFP) scheme where each node runs a Fictitious Play algorithm locally without disclosing its individual payoff function. Then the decision of each node is distributed to all nodes via the server and the corresponding belief tables are updated accordingly at each iteration. The pseudo code of the DFP scheme is given in Algorithm 1. Considering the iterative nature of the algorithm, we carry out the convergence analysis of DFP in Section VI.

Algorithm 1 Decentralized Fictitious Play For NTAG

Input: TK_e, R, N, X, Δ
Output: Task Allocation Strategy for E_x

Initialize: $taskCount \leftarrow newArray[X][N]$, $L \leftarrow newArray[X][N]$, $S \leftarrow newArray[X]$, $converge \leftarrow False$
Assign values to belief table L based on Eq. (7) ▷ Initialize belief table
while $converge \neq True$ **do**
  find $h \leftarrow$ best strategy for E_x based on Eq. (5) and (6)
  $S[x] \leftarrow h$, $taskCount[x][h] \leftarrow taskCount[x][h] + 1$
  send h to server, receive S' of current strategy profile of all nodes
  for all $x' \neq x, 1 \leq x' \leq X$ **do**
    $taskCount[x'][S'[x']] \leftarrow taskCount[x'][S'[x']] + 1$
  end for
  if $S == S'$ **then** ▷ Check convergence
    $converge = True$
  else
    $S \leftarrow S'$
    for all $1 \leq x \leq X$ **do**
      for all $1 \leq i \leq N$ **do**
        $L[x][i] \leftarrow \frac{taskCount[x][i]}{\sum_{h=1}^{N} taskCount[x][i]}$ ▷ Update belief table
      end for
    end for
  end if
end while
Return $S[x]$

C. Dynamic Incentive Mechanism (DIM)

In the previous subsections, we have discussed how to meet the objectives edge nodes by maximizing the payoffs using a NTAG model. In this subsection, we propose a Dynamic Incentive Mechanism (DIM) scheme to meet the QoS objective of the application by updating the reward function of each task in a way that minimizes the data quality loss $\mathcal{L}$ of the application. In particular, we define task reward R_i for task τ_i as a linear combination of data size v_i and task criticality p_i:

$$R_i = \frac{\alpha_1}{\alpha_1 + \alpha_2} v_i + \frac{\alpha_2}{\alpha_1 + \alpha_2} p_i \tag{8}$$

where α_1 and α_2 are the weighting factors. The intuition is that the edge nodes might be unwilling to pick the data-intensive tasks to avoid energy consumption (thus higher incentive is necessary). Moreover, tasks with higher criticality deserve higher rewards. The problem is how to dynamically adjust the reward function to meet the QoS objectives of the application given a fixed budget of incentives.

We adopt the exponential weights algorithm [37] to address the above problem. In particular, we assume there exist two "experts" who vote for the importance of the two factors (i.e., whether "data size" or "criticality" is more important) in the reward function. Each expert has his/her weight (i.e., α_1 and α_2, respectively) on their votes. Based on the experts' votes, the server calculates the reward based on Equation (8). After the tasks are executed, the server can observe which tasks missed the deadline and analyze the data quality loss $\mathcal{L}$. Such data quality loss is re-attributed to each expert as *expert loss* to identify to what extent an expert's vote could potentially cause the data quality loss:

$$\begin{aligned}
\iota_1 &= \frac{\sum_{i=1}^{N} p_i * (1 - \delta_i)}{\sum_{i=1}^{N} (1 - \delta_i)} - \frac{\sum_{i=1}^{N} p_i * \delta_i}{\sum_{i=1}^{N} \delta_i}, \sum_{i=1}^{N} \delta_i \neq 0 \ or \ N \\
\iota_2 &= \frac{\sum_{i=1}^{N} v_i * (1 - \delta_i)}{\sum_{i=1}^{N} (1 - \delta_i)} - \frac{\sum_{i=1}^{N} v_i * \delta_i}{\sum_{i=1}^{N} \delta_i}, \sum_{i=1}^{N} \delta_i \neq 0 \ or \ N
\end{aligned} \tag{9}$$

where ι_1 and ι_2 are the loss functions for criticality and data size factors respectively. δ_i is defined as whether a task τ_i meets the deadline or not (in Section III). In the above equation, we use the average criticality of the tasks that meet the deadlines (i.e., $\frac{\sum_{i=1}^{N} p_i * \delta_i}{\sum_{i=1}^{N} \delta_i}$) as a set point and compare it with the average criticality score of the tasks that missed the deadlines ($\frac{\sum_{i=1}^{N} p_i * (1-\delta_i)}{\sum_{i=1}^{N} (1-\delta_i)}$). Same idea applies for the data size factor. The parameters α_1 and α_2 are tuned based on the comparisons to minimize the loss functions. In the boundary case where all tasks miss the deadline (i.e.,

$\sum_{i=1}^{N} \delta_i = 0$), we double the total amount of rewards. If all tasks meet the deadline (i.e., $\sum_{i=1}^{N} \delta_i = N$), we keep the current configuration. Based on each expert's loss and a tunable learning parameter η, the expert weights are updated:

$$\alpha_i^{new} = \alpha_i^{old} * e^{-\eta \iota_i}, \quad 1 \leq i \leq 2 \tag{10}$$

The pseudo code of the DIM is in Algorithm 2.

Algorithm 2 Dynamic Incentive Update with EWA

Input: TK_e, R, N, X, Δ m, M, η
Output: updated α_1, α_2

if $m == 1$ **then**
  return $\alpha_1 = 0.5$, $\alpha_2 = 0.5$
end if
Perform task allocation based on Alg. 1 and execute tasks
Observe data quality loss $\mathcal{L}$
for all $i, 1 \leq i \leq 2$ **do**
  Calculate ι_i based on Eq. (9)
  Update weight α_i based on Eq. (10)
end for
return (α_1, α_2)

V. Implementation and Experiment Platform

In this section, we present the implementation and experiment platform of the BGTA framework. The prototype of the BGTA framework consists of a remote server and three embedded boards (which will be explained in details in the following subsections). A snapshot of the experiment platform is shown in Figure 3.

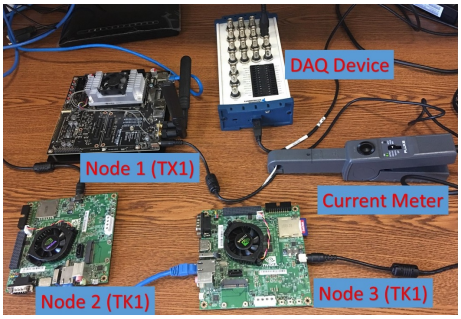

Figure 3. BGTA Implementation

A. Hardware Platform

In our prototype system, the server is a dedicated workstation with Intel Xeon E5-2620 v4 Processor and 16GB of RAM. The edge nodes consist of one Nvidia Jetson TX1 [38] node and two Nvidia Jetson TK1 [39] nodes to emulate mobile platforms with heterogeneous energy profiles and computation capabilities. These two types of edge nodes are commonly used in portable computers, UAVs, and autonomous cars.

B. Energy Measurement

Energy profiling and monitoring are needed in our system. We use FLUKE AC/DC current clamp meters [40] to monitor real-time current signal of each edge node and capture the current data using a National Instruments USB-6216 Data Acquisition (DAQ) system [41]. We then multiply the current with the default voltage (12 V for TK1 and 19 V for TX1) to obtain the real-time power consumption of each edge node.

C. System Modules and Protocol

Server Module: The server module consists of a *server main program* and a *server task executable*. The server main program performs the following tasks: i) periodically define a set of edge tasks. Each task specifies what data to collect and which algorithm to use for data processing; ii) send task information to edge nodes via UDP sockets; iii) receive outputs from the three edge nodes and call server task executable to aggregate results; iv) run DIM to adjust rewards for each task.

Edge Modules: On each edge node, we develop two modules under the BGTA framework: the *edge main program* and the *edge task executable*. The edge main program performs the following tasks: i) receive periodic edge task information from the server; ii) run DFP to select the edge tasks it wants to execute; iii) run Dynamic Voltage Frequency Scaling (DVFS) [27] and execute edge tasks to further optimize energy savings on edge nodes; iv) send results back to the server.

VI. Evaluation

Based on the experiment platform described in the previous section, we conducted extensive evaluation. Below, we first present the basic hardware measurement data and baseline approaches for comparison. We then present the evaluation results for the BGTA scheme using two real world social sensing applications: *Truth Discovery* and *Spatial-temporal Inference*.

A. Experiment Setup

1) Baselines: We choose the following representative task allocation schemes from recent literature as baselines for comparison.

- **Congestion (COG)**: A congestion game based edge computing task allocation scheme where tasks are modeled as resources and the reward of a task is monotonically decreasing as more edge nodes claiming that task [24].
- **Mixed Integer Linear Programming (MILP)**: A top-down task allocation scheme with cooperative edge nodes using MILP with the objective to minimize deadline miss rate [7]. The solver is running on the server.
- **Greedy-Max Reward (GMXR)**: A greedy task allocation scheme where an edge node always selects the task with the highest reward.
- **Greedy-Min Cost (GMNC)**: A greedy task allocation scheme where an edge node always selects the task with the least energy cost.

Note that the MILP scheme is a top-down based approach that requires the server to estimate the status (e.g., background utilization and frequency) of the edge nodes. In our experiment, we dynamically adjust the background utilization estimation and find the best setting for MILP. The server assumes default operational frequency (TK1 at 2.065GHz and TX1 at 1.734GHz) for all edge nodes for the MILP baseline. Given the limited literature on non-cooperative task allocation schemes in edge computing, we propose two heuristic bottom-up algorithms (namely GMXR and GMNC) as additional baselines. To ensure a fair comparison, we applied the DVFS as

the processor level scheduling scheme for energy optimization on edge nodes in *all compared schemes*.

B. Case Study 1: Truth Discovery

The first real world case study is Truth Discovery in social sensing where the goal is to identify truthful claims from massive noisy social sensing data without knowing the reliability of data sources *a priori* [42]. For example, in the Twitter dataset that we collected, a claim is a factual statement about an event (e.g., "3 people died during the Boston Bombing attack") and a source is a Twitter user. In our evaluation, we implement a well-known truth discovery algorithm using a maximum likelihood estimation approach [12].

To evaluate our BGTA scheme with the truth discovery application, we collected a Twitter dataset [1] related to the Boston Bombing event that happened on April. 15, 2013. During that event, two bombs were detonated near the finish line of the annual Boston Marathon and caused three deaths and injured several hundred others. We divide our dataset into 20 subsets and each subset is further split into hourly intervals, which results in 159 periods and 20 tasks (i.e., one task per subset) per period for evaluation. The edge task and the server task in this case study are defined as follows.

Edge Task: A truth discovery edge tasks i) collects a specific data stream using keywords related to an event of interests (e.g. "number of causality", "suspect escape path") assigned by the server; ii) clusters tweets into multiple claims and label each tweet with the claim number; iii) labels each tweet as "agrees with" or "disagree with" its corresponding claim using sentiment analysis. Both the clustering and sentiment analysis techniques are described in detail in [13].

Server Task: A truth discovery server task receives labeled tweets from edge nodes and performs truth discovery using the algorithm in [12].

For each edge task τ_i, the Criticality Score p_i is computed based on the prior knowledge of the importance of the claims. We manually grade each claim with a score of 0 (not important), 5 (medium important) or 10 (very important) via a majority voting of three graders. An example of criticality score assignment is listed in Table I.

Table I
TRUTH DISCOVERY CRITICALITY ASSIGNMENT EXAMPLE

Claim	Criticality Score
Second bombing suspect captured in Watertown	10
Veteran looks to donate prosthetic leg to victims	5
Traffic accident on #Cairo ring	0

Below, we present the evaluation results regarding i) QoS of the application; ii) energy consumption and payoff of edge nodes; iii) convergence of BGTA.

1) Deadline Hit Rate and QoS (Application (Server) Side): The first set of experiments mainly focus on how the objective is achieved from the application side. We first evaluate all the schemes in terms of the deadline hit rate (DHR) and data quality loss (DQL). The results of DHR are shown in Figure 4. We observe that our scheme significantly outperforms other baselines by having higher DHRs and is the first to reach 100% DHR as the deadline increases. The reason is that BGTA explicitly models the dynamic status of the edge nodes (e.g., computation capability and energy profile) and allocates tasks more effectively based on the node status.

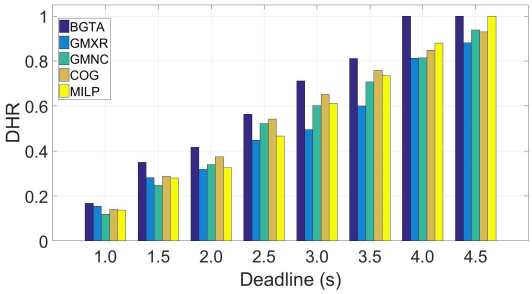

Figure 4. Truth Discovery Deadline Hit Rate vs. Deadlines

The data quality loss (DQL) $\mathcal{L}$ is defined in Equation (4) and represents the QoS loss of the truth discovery application. The results are reported in Figure 5. We observe that our BGTA scheme has the least amount of data quality loss compared to other baselines at all deadlines. We attribute such performance gain to our dynamic incentive mechanism that incentivizes the edge nodes to pick tasks with higher criticality while considering their own computation capabilities.

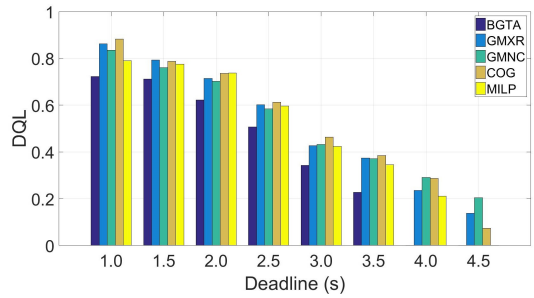

Figure 5. Truth Discovery Data Quality Loss vs. Deadlines

2) Energy Consumption and Payoff (Edge Node Side): The second set of experiments mainly focus on how the objectives are achieved from the edge node's perspective. In particular, we focus on the energy consumption and the payoff at each edge node. We first evaluate the average power consumption over all periods for each scheme. The results are reported in Table II. We observe that BGTA attains the most overall energy savings compared to COG, GMXR, and MILP. The energy savings of BGTA are achieved by allowing edge nodes with more efficient energy profile to pick more tasks (e.g., potentially data/computation intensive ones). BGTA also has slightly better energy saving than GMNC which minimizes the energy consumption by always picking the tasks with the lowest energy consumption. This is because BGTA pushes the energy consuming tasks to the more energy efficient TX1 node, which results in a higher overall energy saving.

The results of payoffs gained at the edge nodes are shown in Figure 6. We observe that our BGTA scheme provides higher payoffs to the edge nodes compared to other baselines. The

[1] http://apollo.cse.nd.edu/datasets

Table II
TRUTH DISCOVERY AVERAGE POWER

Edge Node	BGTA	GMXR	GMNC	COG	MILP
1 (Jetson TX1)	3.3514	3.2254	3.0409	3.2861	3.1521
2 (Jetson TK1)	2.8893	3.3666	2.9840	3.2827	3.3416
3 (Jetson TK1)	2.8661	3.2469	3.0994	2.9112	3.2975
All	9.1068	9.8389	9.1243	9.4800	9.7912

BGTA scheme achieves this by maximizing the payoffs at edge nodes via reaching the Nash Equilibrium in NTAG. We also observe that TX1 node has the highest payoff due to its superior computing power and energy profile.

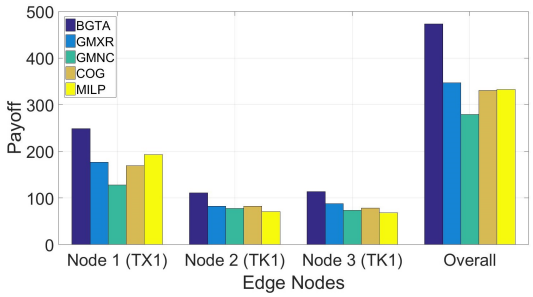

Figure 6. Truth Discovery Individual Node Payoff

3) Convergence of BGTA: We finally evaluate the convergence trend of the BGTA scheme by recording the number of iterations to reach Nash Equilibrium at each period when running the DFP algorithm. The result is shown in Figure 7. We observe that our BGTA scheme converges rapidly with an accumulative probability of 0.92 within 100 iterations. This convergence rate is very efficient compared to other game theoretic models [11], [24] and is achieved by the singleton property that yields a small strategy space of the NTAG model.

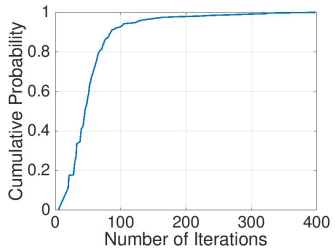

Figure 7. Truth Discovery Cumulative Density Function for Convergence

C. Case Study 2: Spatial-temporal Inference

The second real world case study is Spatial-temporal Inference in social sensing where the goal is to infer/predict the measured variables (e.g., traffic condition, air quality index) in an area by leveraging the measurements collected from social sensors and exploring the spatial-temporal correlation between measured variables. For example, in a mobile crowdsensing project, the participants (together with mobile phones) collect real-time air quality index in different subareas of a city, and the collected data can be used to infer/predict the air quality index of other subareas in a city with no sensor measurements.

To evaluate the performance of BGTA for a real world spatial-temporal inference application, we use a Beijing PM2.5 dataset[2] that contains hourly air quality measurements (PM2.5 Index) from 12 monitoring stations in Beijing, China between May 13, 2014 and Dec 31, 2015. We divide the dataset into hourly intervals and treat each monitoring data station as a subarea. This results in a total of 964 periods and 12 tasks (i.e., one task per subarea). The edge task and the server task in this case study are defined as follows.

Edge Task: A spatial-temporal inference edge task: i) collects the air quality data stream from a specific subarea assigned to the edge node by the server. We assume each edge node can process tasks of any subareas; ii) perform temporal prediction using the ARIMA algorithm [43] to predict/infer the air quality index of the subarea the edge task targets at; iii) send the prediction results to the server.

Server Task: A spatial-temporal inference server task aggregates the temporal predictions from edge tasks and performs KNN spatial inference [44] to infer the air quality index of unmeasured subareas.

Let us assume that task τ_i collects data from subarea z'. We assign Criticality Score as $p_i = \sum_{z=1}^{12} dp(z, z')$ where $dp(z)$ is the spatial dependency between location z and z' based on the physical distance [45]. The score is normalized to a range of 0-10. The intuition is that a task that collects data from a subarea that has stronger dependencies with other subareas is more critical (since the measurements from that subarea can be used to infer other subareas more effectively).

1) Deadline Hit Rate and QoS (Application (Server) Side): We perform similar experiments as those discussed in the previous subsection. In particular, we first evaluate all the schemes in terms of DHR and DQL. The results are shown in Figure 8 and Figure 9 respectively. We observe similar results of BGTA as the previous case study. We also observed that both DHR and DQL scores are better than those in the previous section at the same deadline. The reason is that the ARIMA algorithm is much less computationally intensive than the truth discovery algorithm.

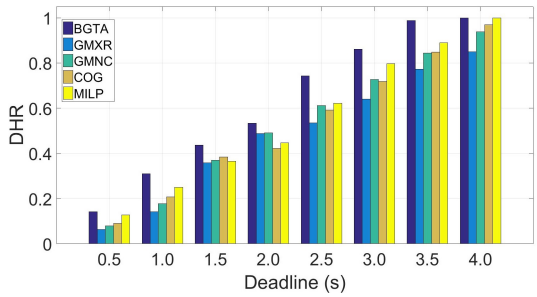

Figure 8. Spatial-temporal Inference Deadline Hit Rate vs. Deadlines

2) Energy Consumption and Payoff (Edge Node Side): The results of energy consumption and payoffs of the edge nodes

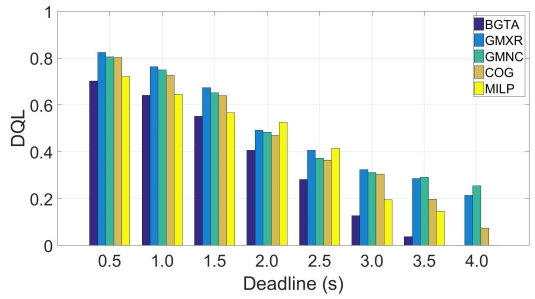

Figure 9. Spatial-temporal Inference Data Quality Loss vs. Deadlines

Table III
SPATIAL-TEMPORAL INFERENCE AVERAGE POWER

Edge Node	BGTA	GMXR	GMNC	COG	MILP
1 (Jetson TX1)	2.9419	3.0578	2.7717	2.9971	2.8446
2 (Jetson TK1)	2.5023	2.9144	2.7521	2.7725	2.9190
3 (Jetson TK1)	2.4324	2.7211	2.5004	2.8919	2.8825
All	7.8766	8.6933	8.0242	8.6615	8.6461

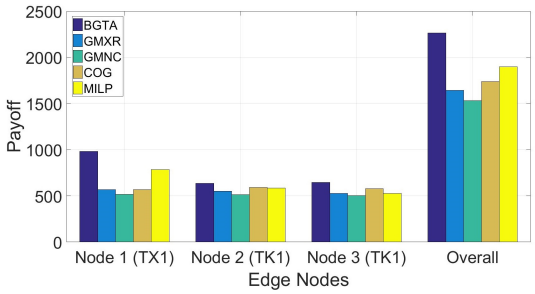

Figure 10. Spatial-temporal Inference Individual Node Payoff

are reported in Table III and Figure 10 respectively. Similar performance gains of BGTA are observed.

3) Convergence of BGTA: The convergence of BGTA in the spatial-temporal inference application is shown in Figure 11. We observe that our BGTA scheme converges rapidly with an accumulative probability of 0.88 within 100 iterations.

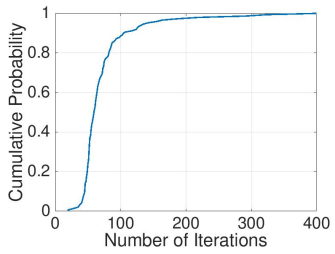

Figure 11. Spatial-temporal Inference Cumulative Density Function for Convergence

VII. LIMITATIONS AND FUTURE WORK

First, we do not explicitly explore the location information of data in the BGTA framework. Such location information could affect the decisions made by the edges node on which tasks to take in some cases (e.g., an edge node might prefer to take a task where the relevant data is closer to the node's location to save communication costs). Our BGTA framework can be easily extended to address such problem by modifying the reward function of a task to explicitly consider the location of the data related to the task (e.g., rewards can be proportional to the distance between edge nodes and data sources).

Second, we assume the edge nodes are not malicious. However, it is interesting to investigate the scenarios where the edge devices can intentionally give wrong results for tasks or send fabricated results without actually running the edge task algorithm. This issue can be addressed by combing BGTA with the verifiable computing techniques where the application collects verifiable results by requiring both execution results and a "proof" of the correct execution of tasks [46].

Third, our current experiment platform consists of a limited number of edge nodes and the scalability aspect of BGTA deserves further investigation. The BGTA has the nice property of guaranteed Nash Equilibrium and is shown to have quick convergence in real world social sensing applications. In the future work, we plan to significantly extend the scale of our experiment platform and also perform additional simulation studies to investigate the scalability of BGTA.

VIII. CONCLUSION

This paper develops a BGTA framework to solve the real time and non-cooperative task allocation problem for social sensing applications in edge computing systems. The BGTA framework addresses two fundamental challenges in solving the task allocation problem, namely *conflicting interests* and *asymmetric and incomplete information*. In particular, we develop a Non-cooperative Task Allocation Game model that allows heterogeneous edge nodes to compete for tasks associated with rewards. We develop a decentralized Fictitious Play algorithm to ensure individual edge nodes make their best decisions with incomplete information of other nodes. We design a Dynamic Incentive component to minimize the data quality loss by adjusting the reward assignments to each edge task. Finally, we implement a prototype of BGTA using the Nvidia Jetson TK1 and TX1 boards. The evaluation results from two real-world social sensing applications demonstrate that BGTA achieves significant performance gains in terms of meeting both the objectives of applications and edge nodes compared to the state-of-the-art baselines.

ACKNOWLEDGMENT

This work is partially supported by the National Science Foundation under Grant No. CBET-1637251, CNS-1566465, IIS-1447795, and CNS-1319904. and Army Research Office under Grant W911NF-17-1-0409. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

[2] https://dataverse.harvard.edu/dataverse/beijing-air

REFERENCES

[1] D. Y. Zhang, D. Wang, and Y. Zhang, "Constraint-aware dynamic truth discovery in big data social media sensing," in *Big Data (Big Data), 2017 IEEE International Conference on*. IEEE, 2017, pp. 57–66.

[2] A. Artikis, M. Weidlich, F. Schnitzler, I. Boutsis, T. Liebig, N. Piatkowski, C. Bockermann, K. Morik, V. Kalogeraki, J. Marecek *et al.*, "Heterogeneous stream processing and crowdsourcing for urban traffic management." in *EDBT*, 2014, pp. 712–723.

[3] X. Chen, E. Santos-Neto, and M. Ripeanu, "Crowdsourcing for on-street smart parking," in *Proceedings of the second ACM international symposium on Design and analysis of intelligent vehicular networks and applications*. ACM, 2012, pp. 1–8.

[4] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Intelligent Systems and Control (ISCO), 2016 10th International Conference on*. IEEE, 2016, pp. 1–8.

[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.

[6] W. E. Walsh and M. P. Wellman, "A market protocol for decentralized task allocation," in *Multi Agent Systems, 1998. Proceedings. International Conference on*. IEEE, 1998, pp. 325–332.

[7] Q. Zhu, H. Zeng, W. Zheng, M. D. Natale, and A. Sangiovanni-Vincentelli, "Optimization of task allocation and priority assignment in hard real-time distributed systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 11, no. 4, p. 85, 2012.

[8] T. A. AlEnawy and H. Aydin, "Energy-aware task allocation for rate monotonic scheduling," in *Real Time and Embedded Technology and Applications Symposium, 2005. RTAS 2005. 11th IEEE*. IEEE, 2005, pp. 213–223.

[9] L. F. Bertuccelli, H.-L. Choi, P. Cho, and J. P. How, "Real-time multi-uav task assignment in dynamic and uncertain environments," 2009.

[10] W. Zhang, E. Bai, H. He, and A. M. Cheng, "Solving energy-aware real-time tasks scheduling problem with shuffled frog leaping algorithm on heterogeneous platforms," *Sensors*, vol. 15, no. 6, pp. 13 778–13 804, 2015.

[11] I. Ahmad, S. Ranka, and S. U. Khan, "Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1–6.

[12] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher, "On truth discovery in social sensing: A maximum likelihood estimation approach," in *Proc. ACM/IEEE 11th Int Information Processing in Sensor Networks (IPSN) Conf*, Apr. 2012, pp. 233–244.

[13] D. Y. Zhang, C. Zheng, D. Wang, D. Thain, X. Mu, G. Madey, and C. Huang, "Towards scalable and dynamic social sensing using a distributed computing framework," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 966–976.

[14] M.-A. Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci, "Computation offloading game for an uav network in mobile edge computing," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[15] Y. Zhang, "A cross-site study of user behavior and privacy perception in social networks," Master's thesis, Purdue University, 2014.

[16] D. Wang, B. K. Szymanski, T. Abdelzaher, H. Ji, and L. Kaplan, "The age of social sensing," *arXiv preprint arXiv:1801.09116*, 2018.

[17] S. Ilarri, O. Wolfson, and T. Delot, "Collaborative sensing for urban transportation." *IEEE Data Eng. Bull.*, vol. 37, no. 4, pp. 3–14, 2014.

[18] H.-P. Hsieh, S.-D. Lin, and Y. Zheng, "Inferring air quality for station location recommendation based on urban big data," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 437–446.

[19] D. Y. Zhang, R. Han, D. Wang, and C. Huang, "On robust truth discovery in sparse social media sensing," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1076–1081.

[20] A. Mtibaa, K. A. Harras, and A. Fahim, "Towards computational offloading in mobile device clouds," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1. IEEE, 2013, pp. 331–338.

[21] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[22] W. Gao, "Opportunistic peer-to-peer mobile cloud computing at the tactical edge," in *Military Communications Conference (MILCOM), 2014 IEEE*. IEEE, 2014, pp. 1614–1620.

[23] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Infocom, 2012 Proceedings IEEE*. IEEE, 2012, pp. 945–953.

[24] D. Liu, L. Khoukhi, and A. Hafid, "Decentralized data offloading for mobile cloud computing based on game theory," in *Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on*. IEEE, 2017, pp. 20–24.

[25] J.-J. Chen and C.-F. Kuo, "Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms," in *Embedded and Real-Time Computing Systems and Applications, 2007. RTCSA 2007. 13th IEEE International Conference on*. IEEE, 2007, pp. 28–38.

[26] H. Su and D. Zhu, "An elastic mixed-criticality task model and its scheduling algorithm," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 147–152.

[27] Y. Ma, T. Chantem, R. P. Dick, and X. S. Hu, "Improving system-level lifetime reliability of multicore soft real-time systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 6, pp. 1895–1905, 2017.

[28] S. Narayana, P. Huang, G. Giannopoulou, L. Thiele, and R. V. Prasad, "Exploring energy saving for mixed-criticality systems on multi-cores," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2016 IEEE*. IEEE, 2016, pp. 1–12.

[29] J. Chen and L. K. John, "Efficient program scheduling for heterogeneous multi-core processors," in *Proceedings of the 46th Annual Design Automation Conference*. ACM, 2009, pp. 927–930.

[30] K. W. Tindell, A. Burns, and A. J. Wellings, "Allocating hard real-time tasks: an np-hard problem made easy," *Real-Time Systems*, vol. 4, no. 2, pp. 145–165, 1992.

[31] H. Ackermann, H. Röglin, and B. Vöcking, "Pure nash equilibria in player-specific and weighted congestion games," in *WINE*. Springer, 2006, pp. 50–61.

[32] O. Rozenfeld and M. Tennenholtz, "Strong and correlated strong equilibria in monotone congestion games," in *WINE*, vol. 4286. Springer, 2006, pp. 74–86.

[33] I. Milchtaich, "Congestion games with player-specific payoff functions," *Games and economic behavior*, vol. 13, no. 1, pp. 111–124, 1996.

[34] X. Vives, "Nash equilibrium with strategic complementarities," *Journal of Mathematical Economics*, vol. 19, no. 3, pp. 305–321, 1990.

[35] E. Campos-Nañez, A. Garcia, and C. Li, "A game-theoretic approach to efficient power management in sensor networks," *Operations Research*, vol. 56, no. 3, pp. 552–561, 2008.

[36] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, vol. 136, no. 2, pp. 215–250, 2002.

[37] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.

[38] Nvidia, "Jetson Tegra X1." [Online]. Available: http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html

[39] ——, "Jetson Tegra K1." [Online]. Available: http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

[40] FLUKE, "80i-110s AC/DC current clamp." [Online]. Available: http://www.fluke.com/fluke/iden/accessories/current-clamps/80i-110s.htm?pid=55352

[41] N. Instruments, "NI USB-6216 BNC." [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/en/nid/207100

[42] D. Wang, T. Abdelzaher, and L. Kaplan, *Social sensing: building reliable systems on unreliable data*. Morgan Kaufmann, 2015.

[43] T. Slini, K. Karatzas, and N. Moussiopoulos, "Statistical analysis of environmental data as the basis of forecasting: an air quality application," *Science of the total environment*, vol. 288, no. 3, pp. 227–237, 2002.

[44] E. G. Dragomir, "Air quality index prediction using k-nearest neighbor technique," *Bulletin of PG University of Ploiesti, Series Mathematics, Informatics, Physics, LXII*, vol. 1, no. 2010, pp. 103–108, 2010.

[45] D. S. Stoffer, *Maximum Likelihood Fitting STARMAX Models to Incomplete Space-time Series Data*. Center for Multivariate Analysis, University of Pittsburgh, 1983.

[46] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Annual Cryptology Conference*. Springer, 2010, pp. 465–482.