Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 5216495, 12 pages
https://doi.org/10.1155/2019/5216495

Research Article

Optimal Task Partition with Delay Requirement in Mobile Crowdsourcing

Linbo Zhai,[1] **Hua Wang**,[2] **and Xiaole Li**[3]

[1]*School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China*
[2]*School of Computer Science and Technology, Shandong University, Jinan 250100, China*
[3]*School of Information Science and Engineering, Linyi University, Linyi 276000, China*

Correspondence should be addressed to Linbo Zhai; zhai@mail.sdu.edu.cn and Xiaole Li; leo0539@163.com

Received 13 May 2019; Revised 15 July 2019; Accepted 14 August 2019; Published 12 September 2019

Guest Editor: Nhu-Ngoc Dao

Mobile crowdsourcing takes advantage of mobile devices such as smart phones and tablets to process data for a lot of applications (e.g., geotagging for mobile touring guiding monitoring and spectrum sensing). In this paper, we propose a mobile crowdsourcing paradigm to make a task requester exploit encountered mobile workers for high-quality results. Since a task may be too complex for a single worker, it is necessary for a task requester to divide a complex task into several parts so that a mobile worker can finish a part of the task easily. We describe the task crowdsourcing process and propose the worker arrival model and task model. Furthermore, the probability that all parts of the complicated task are executed by mobile workers is introduced to evaluate the result of task crowdsourcing. Based on these models, considering computing capacity and rewards for mobile workers, we formulate a task partition problem to maximize the introduced probability which is used to evaluate the result of task crowdsourcing. Then, using a Markov chain, a task partition policy is designed for the task requester to realize high-quality mobile crowdsourcing. With this task partition policy, the task requester is able to divide the complicated task into precise number of parts based on mobile workers' arrival, and the probability that the total parts are executed by mobile workers is maximized. Also, the invalid number of task assignment attempts is analyzed accurately, which is helpful to evaluate the resource consumption of requesters due to probing potential workers. Simulations show that our task partition policy improves the results of task crowdsourcing.

1. Introduction

In recent years, the proliferation of crowdsourcing has shown significant potentials for many application areas. Crowdsourcing, a novel task-solving paradigm, means that human workers are recruited to solve complicated tasks. Extensive researchers are attracted to pay attention to crowdsourcing due to its success about human intrinsic applications.

Early successful examples are Wikipedia, Yahoo! Answers, and Yelp. As the great potential of crowdsourcing is realized in recent years, several general-purpose platforms, including oDesk and Amazon Mechanical Turks (AMT), make crowdsourcing more manageable and powerful. These online systems occur to make requesters define tasks and human workers execute them with rewards. For many online crowdsourcing systems, the common issue is inefficiency. For instance, no more than 15% tasks in Amazon mTurk system could be finished within an hour [1].

On the other hand, recent years witness the remarkable proliferation of intelligent mobile devices (e.g., smart phones and tablets) and the sharp growth of mobile-broadband services including data sharing and synchronization ultra-high-resolution video streaming and virtual and augmented reality. All these services continue to drive the demand for higher data rates and lead to crowdsourcing application in Internet-of-Things (IoT), where pervasive interconnected smart objects cooperates together to reach multiple goals. IoT technologies can effectively promote the interactions between environments and the human and enhance the reliability and efficiency of smart cities [2–7].

With the extensive use of mobile devices such as tablets and smart phones, mobile crowdsourcing systems (MCS) is a feasible solution to complete delay-sensitive tasks such as

geotagging for mobile touring guiding monitoring and local parking space searching which are inevitable in smart cities. To improve the task-executing efficiency, a user tends to assign such delay-sensitive tasks to mobile devices to complete them and collect executing results through the wireless communication system, which is called mobile crowdsourcing. We should notice that these task-executing results must come back within delay requirement. However, mobile devices, which are portable but with less computing ability, are difficult to complete complex tasks within delay requirement. Hence, this is challenging to mobile devices and motivates us to develop a mobile crowdsourcing policy to divide large tasks into several small parts suitable for mobile devices to execute.

In this paper, we propose a mobile crowdsourcing paradigm, dividing a complicated task into small pieces and assigning small subtasks to mobile devices, to obtain high-quality executing results which means mobile devices execute as many subtasks as possible. We design the recruited process and present the worker arrival model and task model. Furthermore, the probability that all parts of the complicated task are executed by mobile workers is introduced to evaluate the result of task crowdsourcing. The higher probability means the result of the task crowdsourcing is better. According to these models, we take into consideration other factors such as the complexity and rewards of tasks which will influence mobile workers' execution and formulate a task partition problem to maximize the probability that all parts of the complicated task are executed by mobile workers. Then, using the Markov chain, we derive a task division policy to realize high-quality results, and the invalid number of task assignment attempts is analyzed accurately based on probability generating function. Simulations show that our task partition policy improves the results of task crowdsourcing.

In this paper, we study the task partition problem of crowdsourcing process in the wireless system consisting of many mobile devices. The main contributions are summarized as follows:

(i) Considering the mobility and capacity of mobile workers, we propose a mobile crowdsourcing paradigm to divide a complicated task into multiple subtasks and assign these subtasks to mobile workers. The mobile worker decides to execute such a subtask or not based on its computing capability and corresponding requirements.

(ii) To describe the attributes of each task, the task model is proposed to denote task load, delay requirement, and monetary reward to the mobile workers. Moreover, an arrival model of mobile workers is proposed to describe the encountered process between the requester and the mobile worker. Based on mobile worker features and task attributes, we establish the system model and formulate a task partition problem to maximize the probability that all subtasks are executed by mobile workers and guarantee the result of task crowdsourcing.

(iii) To realize the high-quality results of task crowdsourcing, using a Markov chain, the state transitions, denoting the number of subtasks executed by mobile workers, can be analyzed. Based on these state transitions within limited period, caused by delay requirement, the optimal task partition can be obtained. Then, according to the optimal task partition, the invalid number of subtask assignment is analyzed accurately with probability generating function in the total crowdsourcing process.

(iv) Simulation results show our proposed policy, compared to fix partition policy and the adaptive scheme, approaches the optimal solutions.

The rest of the paper is organized as follows. In Section 2, a review about related works is provided. In Section 3, we describe the task crowdsourcing process and propose the system model. In Section 4, a Markov chain is developed to solve the task partition problem and the number of invalid attempts is analyzed. In Section 5, the proposed algorithm is evaluated with simulation results. Finally, conclusions are shown in Section 6.

2. Related Works

Human computation has been executed for many centuries. Specifically, if a "human" serves to "compute," there will be a human computation which can be observed. This is the reason why there is a history of Human Computation, which is obviously longer than that of the electronic computer. With the rapid development of Internet web service, especially those facilitating online labor recruiting and managing (e.g., oDesk and Amazon MTurk), human computation begins to experience a new era where the sources of human are not designated exerts or employees but extended to a vast pool of crowds instead. This type of outsourcing to crowds, named crowdsourcing, is receiving countless success in several areas such as logistics, fund raising, monitoring, and so on.

With the extensive use of mobile devices such as tablets and smart phones, the new hybrid architecture occurs to support a massive ad hoc crowd which is composed of distributed mobile nodes and a massive social network around a smart city environment [8]. Therefore, mobile crowdsourcing can be utilized in IoT to complete real-time tasks and improve task efficiency (e.g., localization service, spectrum sensing, and environmental monitoring).

For indoor localization systems, a major bottleneck is the tradeoff between both localization accuracy and site survey costs. In [9], a probabilistic radio map construction method is propose with crowdsourcing collection, taking into account both accuracy and survey costs. In order to obtain the need for location labels, Jung et al. [10] propose the unsupervised learning method to calibrate a localization model based on a global-local optimization scheme. In this hybrid scheme, an efficient global-local interaction reduces the task complexity drastically. In [11], a location-aware infrastructure is proposed to combine a broad sensing layer, centralized cloud federation support, and edge computing. With the sensing

capability of mobile devices, users can obtain the crowd sensing services in IoT. Credible interaction issues among mobile users, however, are still hard problems. Considering the credible interaction, An et al. focus on assigning the crowdsourcing sensing tasks [12]. In the field of environmental monitoring, a path planning approach is introduced for crowd evacuation in buildings [13]. Besides, there are some related researches on E-healthcare service for the crowdsourcing IoT and crowdsourcing industrial-IoT (IIoT) applications [14, 15].

To improve the executing efficiency, a complicated task is often divided into smaller pieces in crowdsourcing [16]. Then, these small pieces can be assigned to a group of undefined workers [17]. In many existing studies, there is a central service entity which not only collects the information of both workers and requesters but also performs the task-worker assignment to optimize a global utility [18, 19]. If someone hopes to use a crowdsourcing system [20, 21], inevitably, corresponding fee must be paid for using the centralized server.

Due to the self-organized nature, requesters in mobile networks, however, do not obtain worker information in advance. Therefore, requesters have to probe worker ability and make sequential recruitment decision. This problem motivates us to design a task partition policy for mobile worker recruitment.

In [22], Tuncay and Helmy use the location visiting frequency to reflect the worker ability and present a gradient ascend policy which assign the task to a user having higher visiting frequency in the destination area. However, the objective, similar to the traditional routing and data dissemination, is to arrive at the destination as fast as possible with minimum replication overhead. In [23, 24], Chang and Gong analyze the schemes for allocating work segments among participants opportunistically. When determining users' workloads, the schemes take into account contacting delay, acceptance probability, computing speed, and the existence of resource competition works. The purpose is to minimize the task makespan by partitioning the total task into suitable subtasks corresponding to the encountered worker computation power. In [25], Pu et al. use an online algorithm to maximize the task service quality based on worker's interest preference. Focusing on device-to-device networks, the authors study the recruitment problem when the crowdsourcing is delay-sensitive [26]. In [27], Tong et al. study a large-scale crowdsourcing task which consists of thousands or millions of atomic tasks. To the best of our knowledge, little work has been done on jointly considering the mobile worker features and task partition.

3. System Model

In this section, at first, the task crowdsourcing process is described. Then, the system model, involving task attributes and mobile user features is developed, and the task partition problem is formulated to achieve optimal results.

3.1. Recruitment Description. A mobile user having a task is a requester. The requester can self-organize task crowdsourcing by recruiting several encountered mobile users who are workers in real-time. A mobile user encountering another user means they are close to establish a device-to-device (D2D) link. The term of "close" means the distance between two mobile users does not exceed WiFi-direct distance. The recruitment is described as follows.

When a mobile user (requester) launches a task, the requester is invoked to recruit some encountered mobile users (workers) in real-time. Since a complicated task is difficult for a worker to complete in time, the requester will divide the complex task into several subtasks and recruit corresponding number of workers to complete those subtasks. The requester sends attributes of a subtask including the reward, subtask load, and delay requirement to an arrived worker. The worker decides to execute this subtask or not based on available computing capacity, remaining energy, subtask reward, and delay requirement. If the worker decides to accept this subtask which means it can complete the subtask within delay requirement, its worker ability is set to 1. Otherwise, it is set to zero. Then, the worker ability value is sent to the requester. Finally, the requester decides to recruit the worker or not based on its worker ability.

If the requester decides to recruit the worker, the detailed content of a subtask is sent to the worker. During the subtask execution, it is not necessary for the requester and the worker to connect with each other all the time. After the subtask is finished by the worker, the corresponding subtask execution result will be sent back to the requester by a D2D link or cellular link satisfying the delay requirement. Once the requester receives the result in time, the subtask reward will be given to the worker. Otherwise, if the delay requirement is not satisfied, the requester thinks the worker is fraudulent and does not give the subtask reward to the worker.

The key rationale of the recruiting model is that opportunistic encounters of mobile devices are sufficient and prevalent in modern society [28], which offers lots of opportunities to exploit nearby intelligent devices [29]. If a mobile user has a complex task, it can self-organize its task crowdsourcing by leveraging many encountered mobile users in real-time, and fast response can be realized by interacting with intelligent workers in proximity directly. Furthermore, many mobile crowdsourcing tasks will require location-aware knowledge and information (e.g., mobile touring guiding monitoring and local parking space searching). Hence, nearby intelligent workers are more qualified to execute them than the online workers [22]. In addition, compared to the new emerging paradigm "cyber foraging" about mobile computing, our framework shares the similar spirit so that mobile users can exploit nearby intelligent devices to facilitate their computational task processing [30].

Figure 1 illustrates the procedure of mobile crowdsourcing. A mobile user (requester R), launching a mobile crowdsourcing task, moves around within the network and encounters a mobile user (worker W_1). Requester R divides this task and sends subtask attributes to worker W_1 through a D2D link at position 1. Worker W_1 decides to execute the subtask and returns the worker ability to requester R. Then,

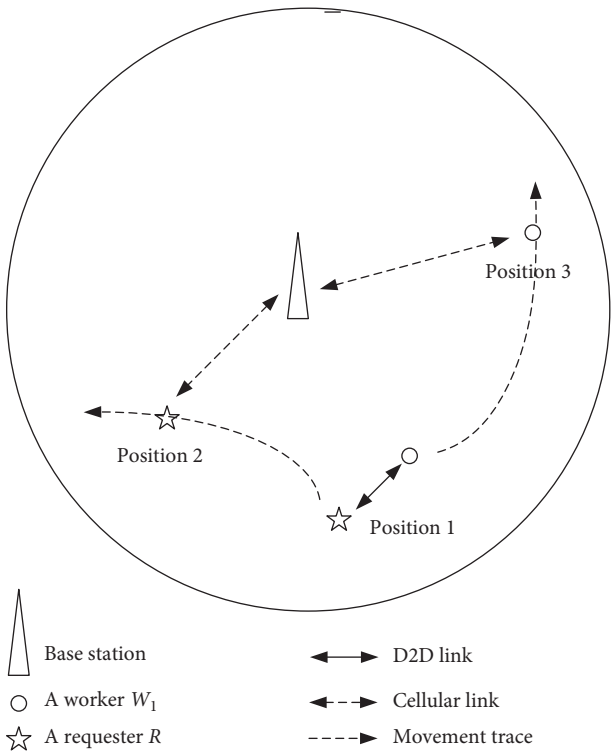

Figure 1: Process of mobile crowdsourcing.

requester R sends the subtask content to worker W_1. During the period of subtask execution, worker W_1 and requester R may continue to move and do not need to connect with each other. When worker W_1 finishes the subtask, worker W_1 and requester R are at position 3 and position 2, respectively. The distance between worker W_1 and requester R is too long to establish D2D link. Therefore, worker W_1 sends the result to requester R by a cellular link, the wireless transmission through mobile communication system [28]. That means worker W_1 sends the result to the base station at first. Then, the base station relays the result to requester R. After that, the subtask reward is granted to worker W_1 by requester R.

3.2. System Model

3.2.1. Worker Arrival Model. We use K to denote the set of potential mobile workers who may execute the task. Considering worker mobility, the inter-encounter time of a requester r and a worker w is important. In our system, it is not necessary for the requester r to know the arrival rate λ_{rw} of each worker w. The total arrival rate $\lambda_r = \sum_{w \in k} \lambda_{rw}$, obtained by calculating the number of encountered workers per time unit, is adopted instead. The current value of total arrival rate can be estimated based on the value during the recent time units. When the total arrival rate is obtained, the average inter-encounter time of a requester r and a worker w is $1/\lambda_r$.

3.2.2. Task Model. A mobile requester can describe a task i by a set of attributes, $<Ta_i, D_i, R_i>$, where Ta_i denotes the task load of task i; Di represents delay requirement of task i within which a worker must return the result; R_i denotes the total reward to workers after the results are returned within D_i.

There are three parameters in the task model. Task load and delay requirement are determined by the category of the corresponding task such as content creation or information finding [31]. For the reward, similar to online systems, this model uses a common posted price method where the workers are provided the explicit price offer [32].

3.2.3. Task Partition Formulation. Since a task may be too complex for a mobile worker with less computing ability to complete in time, the task can be divided into many subtasks by the task requester. Then, each subtask of the task is suitable for a worker to finish.

Assume that task i is averagely divided into N subtasks. Then, the load of each subtask is

$$STa_i(N) = \frac{Ta_i}{N}. \tag{1}$$

The reward for a worker finishing a subtask is

$$R_i(N) = \frac{R_i}{N}. \tag{2}$$

The requester sends the subtask attributes including $R_i(N)$, $STa_i(N)$, and delay requirement D_i to a worker w. The worker w evaluates whether $STa_i(N)$ can be completed within D_i with satisfying reward $R_i(N)$.

Let E_w denote the remaining energy of worker w, B_w denote the computing capacity of worker w, and P_{iw} denote the probability that worker w accepts the subtask of task i. Without loss of generality, it is defined that the probability that a worker accepts a subtask is related to $R_i(N)$, E_w, B_w, and $STa_i(N)/D_i$. Intuitively, $R_i(N)$, E_w, and B_w will have positive impact on the accepting probability, while $STa_i(N)/D_i$ will have negative impact on the accepting probability. However, the accepting probability is not simply defined as a function which is proportional to $R_i(N)$, E_w, and B_w, and inversely proportional to $STa_i(N)/D_i$ since the real case is more complicated. Let RE denote the expectation reward of a mobile worker. If a mobile worker can gain the reward $R_i(N)$ exceeding its expectation RE, it will pay more attention to the ratio between the subtask load $STa_i(N)/D_i$ and its computing capacity B_w. Otherwise, if the reward it can gain is less than its expectation RE, it will focus on the real reward it can gain. Additionally, a worker's remaining energy E_w also influences the probability of accepting the subtask. On the one hand, the larger value of E_w will result in the higher accepting probability. On the other hand, the value of E_w will influence the pace of the probability variety. When the remaining energy of a mobile worker is high, it is sensitive and its accepting probability will change drastically as other parameters including $R_i(N)$, B_w, and $STa_i(N)/D_i$ change. When a mobile worker has the low remaining energy, its accepting probability will change slowly as other parameters change.

Based on the aforementioned description, the accepting probability should be the piecewise function. When the reward $R_i(N)$ exceeds its expectation RE, $STa_i(N)/D_iB_w$ has

more influence on the probability and the probability will decrease as $STa_i(N)/D_iB_w$ increases. When the reward $R_i(N)$ is lower than its expectation RE, the reward will have greater influence on the probability and the probability will increase as RE increases. To clearly reflect the influence of $STa_i(N)/D_iB_w$ and $R_i(N)$ on the trend of the function variety, we use the square of them rather than their original forms in the piecewise function. Besides, the value of the accepting probability should be in the range (0, 1). Hence, the exponential function is used to guarantee the accepting probability P_{iw} in this range. Then, P_{iw} can be defined as

$$P_{iw} = \begin{cases} \exp\left(-C\left(\dfrac{STa_i(N)}{D_iB_w}\right)^2 \dfrac{1}{R_i(N)E_w}\right) R_i(N) \geq R_E, \\ \exp\left(-C\left(\dfrac{STa_i(N)}{D_iB_w}\right) \dfrac{1}{R_i(N)^2E_w}\right) R_i(N) < R_E, \end{cases} \tag{3}$$

where C is an adjustable constant to control the changing speed of accepting probability.

Using (1) and (2), the probability P_{iw} in Equation (3) can be rewritten as

$$P_{iw} = \begin{cases} \exp\left(-C\dfrac{1}{N}\left(\dfrac{Ta_i}{D_iB_w}\right)^2 \dfrac{1}{R_iE_w}\right) N \leq \dfrac{R_i}{R_E}, \\ \exp\left(-CN\left(\dfrac{Ta_i}{D_iB_w}\right) \dfrac{1}{R_i^2E_w}\right) N > \dfrac{R_i}{R_E}. \end{cases} \tag{4}$$

Since accepting probability P_{iw} is the piecewise function based on the relationship between $R_i(N)$ and RE and Equation (3) only describes the relationship between $R_i(N)$ and RE, using equation (2), we can obtain the relationship between R_i and RE. From Equation (4), it can be observed that the accepting probability P_{iw} increases with the growth of subtask number N when N is less than R_i/RE. On the contrary, the accepting probability P_{iw} decreases with the growth of the subtask number N when N is higher than R_i/RE. This can be explained as follows. When the subtask number N is smaller, the worker is able to gain reward more than its expectation RE. As N increases, a subtask load decreases and the worker is more likely to accept such a subtask. After N exceeds R_i/RE, the worker is not able to achieve its expected reward RE and it mainly focuses on the current reward. As N increases, the current reward of a subtask decreases and the worker is less likely to accept such a subtask.

Additionally, each worker needs some time to complete a subtask. The duration is determined by $STa_i(N)$ and the worker computing capacity. Let $B = \min\{B_w \mid w \in K\}$ denote the minimum computing capacity of potential workers, where K is the set of total mobile workers. Then, the maximum duration for executing a subtask is derived as

$$SD_i = \frac{STa_i(N)}{B}. \tag{5}$$

Therefore, the requester should assign subtasks to workers within $D_i - SD_i$. Otherwise, the delay requirement cannot be guaranteed. Let T_i $(T_i \leq D_i - SD_i)$ denote the duration of subtask assignment. To maximize the service quality, the requester should find as many workers executing subtasks as possible within T_i. Thus, $\lambda_r T_i$ denotes the total number of workers which a requester encounters within T_i. Since workers may accepting the subtasks following P_{iw}, the number of accepting workers must be less than $\lambda_r T_i$. Let n denote the number of workers accepting subtasks. If n exceeds the total number of subtasks, which is denoted by N, all subtasks are executed by workers and the service quality is guaranteed.

To evaluate the result of task crowdsourcing, the probability $P(n \geq N)$, denoting that all subtasks of a complicated task are executed by mobile workers, is introduced. The higher the probability that n exceeds N is, the better service quality is.

The task partition policy for task i is to divide task i into N subtasks which are suitable for workers to execute. The objective is to obtain the optimal task partition N^* to maximize the probability that all subtasks are executed by workers within T_i. Therefore, the optimal task partition problem can be formulated as

$$\begin{aligned} & N^* \triangleq \arg\max_N P(n \geq N), \\ \text{subject to} \quad & T_i \leq D_i - SD_i. \end{aligned} \tag{6}$$

4. Partition Policy

In this section, a Markov chain is used to describe the state transitions denoting subtasks assignment and calculate the optimal task partition. Then, using the transition matrix, the unsuccessful number of subtask assignment attempts before the crowdsourcing end can be analyzed. Furthermore, the time complexity of proposed algorithm is analyzed.

4.1. Optimal Task Partition. According to the description in Section 3, for task i, a requester should receive the executing results from encountered workers within delay requirement D_i. Additionally, each worker needs a duration SD_i to complete a subtask. The duration is determined by $STa_i(N)/B$. Therefore, the requester needs to assign subtasks to workers within $D_i - SD_i$ to satisfy the delay requirement of task i.

It is defined that each slot duration is $T_s = 1/\lambda_r$ where λ_r denotes the arrival rate of total mobile workers. Then, a requester encounters one mobile worker in a slot on average. Since each slot duration T_s equals $1/\lambda_r$, $D_i - SD_i$ can be divided into the m slots. We can obtain

$$m = \frac{D_i - SD_i}{T_s}. \tag{7}$$

Figure 2 shows the slot division within $D_i - SD_i$.

To realize the optimal task partition, we use a Markov chain to describe state transitions within task assignment duration $D_i - SD_i$. For task i which is divided into N subtasks averagely, we use S_0, S_1, S_2, ..., S_N to denote $N+1$ states, where S_j $(j \in [0, N])$ means j subtasks have been assigned to workers successfully.

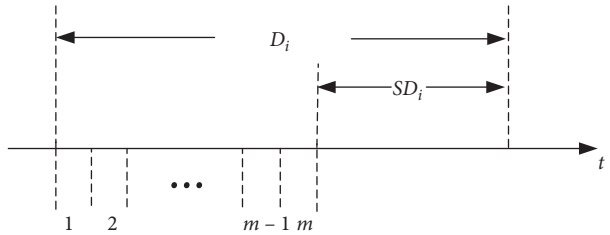

Figure 2: Slot division.

The encountered worker will accept a subtask of task i with the probability P_{iw}, and refuse the subtask with the probability $1 - P_{iw}$. If the encountered worker accepts the subtask, the system moves to new state $S_j + 1$ ($j \le N - 1$) from state S_j. Otherwise, the system still stays at state S_j if the worker refuses the subtask. When the system moves to state SN, which means all subtasks of task i have been accepted by workers, the state will not change no matter the requester encounters new workers or not. Therefore, the one-step transition probability matrix $\mathbf{Q}$, which is a square matrix with order equaling $N + 1$, can be derived as follows:

(i) When the system is currently in the state S_j ($0 \le j \le N - 1$), after one-step transition, the system will move to state $S_j + 1$ with the probability P_{iw} or stay at S_j with the probability $1 - P_{iw}$. Thus for $0 \le j \le N - 1$, $\mathbf{Q}_j, j+1 = P_{iw}$ and $\mathbf{Q}_j, j = 1 - P_{iw}$.

(ii) When the system is currently in the state S_j ($j = N$), the system must stay at S_j after one-step transition. Thus for $j = N$, $\mathbf{Q}_j, j = 1$.

The one-step transition probability matrix $\mathbf{Q}$ is a sparse matrix. We can obtain

$$\mathbf{Q} = \begin{bmatrix} 1 - P_{\text{iw}} P_{iw} & & & & & & \\ & 1 - P_{\text{iw}} P_{iw} & & & & & \\ & & 1 - P_{\text{iw}} P_{iw} & & & & \\ & \cdots & \cdots & \cdots & & & \\ & & & 1 - P_{\text{iw}} P_{iw} & & & \\ & & & & 1 - P_{\text{iw}} P_{iw} & & \\ & & & & & 1 \end{bmatrix}. \tag{8}$$

From the initial slot to the last slot, the system witnesses $m - 1$ transition. The system stays at state S_0 at the initial slot because no subtask is assigned. Hence, the initial distribution is $\eta = [\underbrace{1, 0, \ldots, 0}_{N}]$. Let $P_i(N)$ denote the probability that a requester has assigned total N subtasks of task i successfully within the duration $D_i - SD_i$. Within the delay requirement, there are m slots. We use a Markov chain to describe the state transition from the first slot to the mth slot. Hence, there are $m - 1$ steps. For each step, the transition probability matrix is $\mathbf{Q}$. Therefore, $\mathbf{Q}^{m-1}$ denotes the transition probability matrix of $m - 1$ steps. Then, we can obtain

$$P_i(N) = \eta \mathbf{Q}^{m-1} R, \tag{9}$$

where $R = [0, \ldots, 0, 1]^T$. $P_i(N)$ is related to the task partition number N.

To maximize $P_i(N)$, by solving the Karush–Kuhn–Tucker (KKT) conditions [33], we obtain

$$\frac{\partial}{\partial N} P_i(N) = 0. \tag{10}$$

Then, using (4), (5), (9), and (10), we can derive the optimal task partition N' in theory. Considering the practical limitation of partition number, the optimal task partition N^* is derived as

$$N^* = \min\{N', N_{\max}\}, \tag{11}$$

where $N_{\max}$ denotes the allowable maximum partition number. After N^* is determined, the maximal value $P_i(N^*)$ of successful assignment probability can be calculated based on (9).

For a complicated task, we use a Markov chain and the state transition matrix to obtain the optimal task partition. When the optimal partition N^* is determined, the probability $P_i(N^*)$ that all parts of the complicated task are executed by mobile workers can be obtained. Based on the value of probability $P_i(N^*)$, we know how often this total assignment occurs. The higher value of $P_i(N^*)$ means this total assignment will occur more often. The proposed algorithm for task partition is described as Algorithm 1.

Theorem 1. *The proposed task partition is optimal.*

Proof. Since a complex task is difficult for a mobile device to execute, we divide a complex task into N subtasks and assign each subtask to a mobile device. If each subtask is accepted by a mobile device, the crowdsourcing process is completed. Using a Markov chain, we aim to obtain the optimal value of N to maximize the completed probability of the crowdsourcing process.

At first, the states of the Markov chain are defined. We use $S_0, S_1, S_2, \ldots, S_N$ to denote $N + 1$ states where S_j means j subtasks have been accepted by mobile devices. S_0 means no subtask is assigned while S_N means all subtasks have been assigned successfully. Hence, these states describe how many subtasks are accepted.

Input:
Total mobile workers' arrival rate λ_r
Delay requirement D_i
Task load Ta_i
Monetary reward R_i to a worker
Mobile workers' computing capacity B
Output:
The optimal task partition N^* and the maximal value $P_i(N^*)$
(1) Based on equation (1), $STa_i(N) \longleftarrow$ Each subtask load
(2) Based on equation (3), $P_{iw} \longleftarrow$ The probability that worker w accepts the subtask of task i
(3) With P_{iw}, transition probability matrix $\mathbf{Q}$ is obtained based on Equation (8)
(4) Based on equation (5), $SD_i \longleftarrow$ Duration for each worker to complete a subtask
(5) Based on equation (7), $m \longleftarrow$ The number of slots within $D_i - SD_i$
(6) Within m slots, $P_i(N)$ is obtained based on equation (9)
(7) Using equation (10), $N' \longleftarrow$ the optimal task partition in theory
(8) Using equation (11), $N^* \longleftarrow$ the practical optimal task partition
(9) Based on equation (9) and N^*, $P_i(N^*) \longleftarrow$ the maximal value of successful assignment probability
Return N^* and $P_i(N^*)$

Algorithm 1: Optimal task partition for task i.

If the encountered worker accepts the subtask with the probability P_{iw}, the system moves to new state $S_j + 1$ ($j \le N - 1$) from state S_j. Otherwise, the system still stays at state S_j if the worker refuses the subtask with the probability $1 - P_{iw}$. When the system moves to state S_N, which means all subtasks of task i have been accepted by workers, the state will not change no matter the requester encounters new workers or not. Therefore, we can obtain one-step transition probability matrix $\mathbf{Q}$, shown in equation (8), to describe state transitions.

Within the delay requirement, the crowdsourcing process is completed if the system moves from S_0 to S_N, which means all subtasks are accepted by workers. We use $P_i(N)$, shown in equation (9), to denote this completed probability that the system moves from S_0 to S_N. This probability is determined by the partition value N. Hence, the proper value of N will result in the maximized completed probability. We solve this problem based on equations (10) and (11) and obtain the optimal value N^* to maximize the completed probability, which means optimal partition is obtained. □

4.2. Unsuccessful Assigned Attempts. Before a requester assigns all subtasks successfully, there may be some unsuccessful assignment attempts for the requester because some encountered workers do not accept the subtasks. These invalid attempts consume the requester's resources such as energy and probing time. Based on the optimal task partition N^*, the invalid attempt number can be obtained accurately as follows.

After the optimal task partition N^* is derived, the corresponding one-step transition probability matrix $\mathbf{Q}N^* + 1$ with order equaling $N^* + 1$ is determined. Let U denote the unsuccessful number of subtask assignment attempts before the system reaches state S_{N^*}. To derive the average unsuccessful attempt number $E[U]$, the invalid attempts should be distinguished from the state transition. Thus, we introduce the matrix $\mathbf{Q}N^* + 1(x)$ associated to $\mathbf{Q}N^* + 1$ with dummy variables x. The $\mathbf{Q}N^* + 1(x)$ is defined as follows:

(i) For $0 \le j \le N^* - 1$, $\mathbf{Q}N^* + 1(x)j$, $j+1 = P_{iw}$, and $\mathbf{Q}N^* + 1(x)j$, $j = x(1 - P_{iw})$.
(ii) For $j = N^*$, $\mathbf{Q}N^* + 1(x)j$, $j = 1$.

Note that $\mathbf{Q}N^* + 1(x) = \mathbf{Q}N^* + 1$ if we take $x = 1$. Then, to describe invalid attempts among the state transitions, we can define

$$y(x) = \eta \mathbf{Q}_{N^*+1}(x)^{m-1} R. \tag{12}$$

Let $v(x) = y(x)/P_i(N^*)$. Then we can obtain

$$v(x) = \frac{\eta \mathbf{Q}_{N^*+1}(x)^{m-1} R}{P_i(N^*)}. \tag{13}$$

Considering that $y(x) = P_i(N^*)$ when x equals 1, based on (13), $v(x)$ can also be expressed by

$$\begin{cases} v(x) = \sum_{q=0}^{m-1} f(q) x^q, \\ \sum_{q=0}^{m-1} f(q) = 1, \end{cases} \tag{14}$$

where the coefficient $f(q)$ denotes the probability the system reaches state S_{N^*} with q unsuccessful subtask assignment attempts after $(m-1)$-step transitions from slot 1. So, $v(x)$ is the probability generating function of the unsuccessful number of subtask assignment attempts within $D_i - SD_i$.

Let $E[U]$ be the average unsuccessful number of subtask assignment attempts within $D_i - SD_i$. We can obtain

$$E[U] = \frac{\partial}{\partial x} v(x) \mid_{x=1}. \tag{15}$$

In this section, "dummy variable x" is introduced to calculate the number of unsuccessful assigned attempts.

During the state transition, based on $y(x)$ in (12), all unsuccessful assignments are labeled by the dummy variable x. To calculate the unsuccessful assignment number, we use the properties of generation function. Hence, we use $y(x)/P_i(N^*)$ in (13) to guarantee the coefficient of each term including dummy variable x is in the range (0, 1) based on generation function format. Then, (13) can be rewritten as (14), which is the expression of the probability generating function for the unsuccessful number. Based on the properties of generation function, the average unsuccessful number of subtask assignment attempts is equivalent to the first derivative of $v(x)$ at the point $x = 1$. Here, the invalid number of subtask assignment attempts is analyzed accurately, which is helpful to evaluate the resource consumption of requesters due to probing potential workers.

The proposed algorithm for unsuccessful assignment attempts is described as Algorithm 2.

4.3. Analysis of Time Complexity. The complexity of proposed algorithm is computed as follows.

We first discuss computation complexity of optimal task partition. In line 7 of Algorithm 1, the solution of optimal task partition determines the complexity of Algorithm 1. Considering the allowable maximum partition number $N_{\max}$, the practical partition number cannot exceed this threshold. Meanwhile, the primary computation is matrix multiplication based on (9), and allowable maximum partition number $N_{\max}$ limits the matrix size. Thus, the solution complexity is $O((N_{\max})^{3m})$, where m denotes the number of slots within delay requirement. Based on (7), m is determined by the arrival rate λ_r of total mobile workers and the delay requirement $D_i - SD_i$ of task i. Therefore, computation complexity of Algorithm 1 is $O((N_{\max})^{3m})$, which is determined by the allowable maximum partition number, the total arrival rate, and the delay requirement.

Then, we analyze the computation complexity of unsuccessful assignment attempts. In line 2 of Algorithm 2, the calculation of $y(x)$ determines the complexity of Algorithm 2. Based on slot number m and optimal partition N^* obtained in Algorithm 1, the calculation complexity of Algorithm 2 is $O((N^*)3^m)$. It seems that slot number m and optimal partition N^* determine the computation complexity of Algorithm 2. Actually, since slot number m and optimal partition N^* are the results of Algorithm 1, the complexity of Algorithm 2 is also determined by the allowable maximum partition number, the total arrival rate, and the delay requirement.

5. Simulations

In this section, our proposed task partition policy is evaluated by extensive simulations. Compared to the fixed partition scheme and adaptive scheme in [23], our policy can realize high task service quality.

5.1. Comparison Metric. For efficient comparison, in this section, we adopt two metrics: task service quality and invalid number of subtask assignment attempts.

The first metric, task service quality, is denoted by the ratio between the real completed subtasks and the total subtasks. Obviously, the more subtasks have been finished, the better task service quality can be realized. This metric can clearly illustrate how many subtasks are accepted and executed by mobile workers within the duration $D_i - SD_i$. By using this metric, we can observe different methods lead to various impacts on real completed subtasks.

The second metric calculates unsuccessful number of subtask assignment attempts within delay requirement. During the process of probing potential workers, some workers do not accept subtasks. Thus, these assignment attempts are unsuccessful. This leads to useless resource consumption of requesters and workers. The larger invalid subtask assignment attempts are, the higher invalid resource consumption is. This metric can clearly illustrate how many subtask assignment attempts are not accepted by mobile workers within the duration $D_i - SD_i$. By using this metric, it is helpful for us to understand the resource consumption of requesters and workers in the mobile crowdsourcing system.

The simulation environments are described as follows. As the user mobility model is widely used and validated by research works [34, 35], we use the same parameters of the mobility model as in these studies. The mobile workers come following exponential distribution, and the total arrival rate is denoted by λ_r. Note that our task model fits several mobile crowdsourcing tasks including location-based information finding tasks and content creation tasks. Here, we do the simulation based on the content creation tasks, and all task parameters are the same as in [25]. B, Ta_i, D_i, and R_i are all in the unit of slots. The requester and workers are able to communicate with others by cellular links or D2D links.

5.2. Simulation Results. With the total arrival rate $\lambda_r = 1$, Figure 3 depicts the result of task crowdsourcing as the task load Ta_i increases from 300 to 400. In Figure 3, parameters are set as $D_i = 100$, $R_i = 10$, $E_w = 0.2$, $C = 6$, and $B = 2$. As shown in Figure 3, compared to the fix partition policies, which divide the task into 3 subtasks ($N = 3$) and 8 subtasks ($N = 8$), and the adaptive scheme in [23], our partition policy can achieve higher service quality. As the total task load changes, our policy divides the task into various number of small subtasks to improve the accepting probability of mobile workers, while the fix partition method divides the task into fixed number, leading to the low accepting probability of mobile workers. Also, our partition policy is better than the adaptive scheme because the adaptive scheme does not take into consideration the mobile feature of workers. As the task load Ta_i increases, the service quality decreases because it is difficult for workers with limited computing capacity to complete complex subtasks.

Figure 4 depicts the result of task crowdsourcing as the delay requirement D_i increases from 80 to 100. In Figure 4, parameters are set as follows: $R_i = 10$, $E_w = 0.2$, $C = 6$, $B = 2$, and $Ta_i = 350$. As shown in Figure 4, the service quality increases as the delay requirement D_i increases, because more time is permitted for workers to finish subtasks.

Input:
The probability P_{iw} denoting that a worker accepts the subtask of task
The optimal task partition N^*
The maximal value $P_i(N^*)$ can be obtained
The number m of slots within $D_i - SD_i$
Output:
Average unsuccessful number of subtask assignment attempts $E[U]$
(1) Based on transition probability matrix $\mathbf{Q}$ and the optimal task partition N^* in Algorithm 1, $\mathbf{Q}N^* + 1(x)$ is obtained
(2) Using $\mathbf{Q}N^* + 1(x)$, $y(x)$ is defined to describe invalid attempts based on Equation (12)
(3) Using equations (13) and (14), $y(x)$ is converted to $v(x)$ denoting the probability generating function of unsuccessful attempts
(4) Using Equation (15), average unsuccessful attempts $E[U]$ is obtained
Return $E[U]$

Algorithm 2: Unsuccessful assignment attempts for task i.

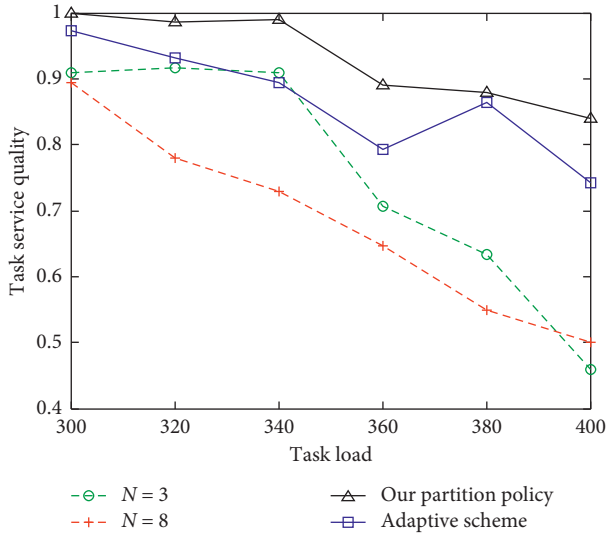

Figure 3: Task service quality varies with task load.

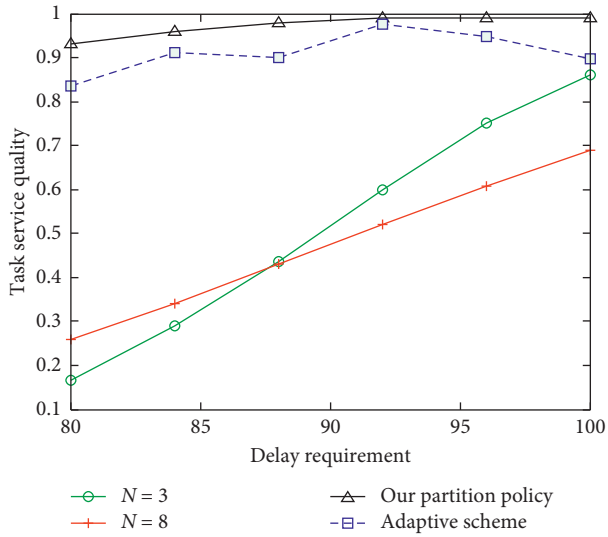

Figure 4: The task service quality varies with delay requirement.

Compared to the fix partition policies ($N = 3$ and 8) and the adaptive scheme, our policy achieves higher quality. Based on the change of delay requirement, our policy adjusts the length of each subtask by task partition to increase the accepting probability of mobile workers, while the fix partition method divides the task into fixed number, which does not consider the accepting probability of mobile workers. Also, our partition policy is better than the adaptive scheme since the adaptive scheme does not take into consideration the mobile feature of workers. Besides, the fix partition policy of $N = 8$ realizes higher service quality than that of $N = 3$ when the delay requirement is small. The policy of $N = 8$ means each subtask is smaller than that of $N = 3$. Intuitively, a smaller subtask is more likely to be completed within strict delay requirement. Hence, the fix partition policy of $N = 8$ realizes higher service quality under small delay requirement. As the delay requirement increases, the completed probability of a bigger subtask also grows. When a bigger subtask is completed, it brings more impact on the service quality. Hence, under the large delay requirement, the fix partition policy of $N = 3$ realizes higher service quality.

Figure 5 depicts the result of task crowdsourcing as the reward R_i increases. In Figure 5, parameters are set as follows: $Ta_i = 350$, $D_i = 80$, $C = 6$, and $B = 2$. As shown in Figure 5, we can see the service quality increases when the task reward R_i increases. The reason is that the probability that workers accept subtask becomes higher with the larger R_i. Our partition policy, even under the lower reward condition, can realize higher task service quality because our policy divides the task into small pieces based on the current reward to increase the worker's accepting probability of each subtask. Furthermore, the fix partition policy of $N = 8$ grows sharply as the reward increases. Initially, the total reward is low and each subtask is given a small reward. Hence, the accepting probability is low and the service quality is small. When the reward increases, each subtask is corresponding to higher reward. For fix partition policy of $N = 8$, the accepting probability increases sharply than others, leading to higher service quality.

When total arrival rate of mobile workers changes, the task service quality of our partition policy also changes. In Figure 6, we set task load $Ta_i = 350$ and 400, respectively. Other parameters are set as follows: $Ri = 10$, $Di = 80$, $C = 6$, and $B = 2$. As shown in Figure 6, we can see the service quality increases when there is higher total arrival rate of

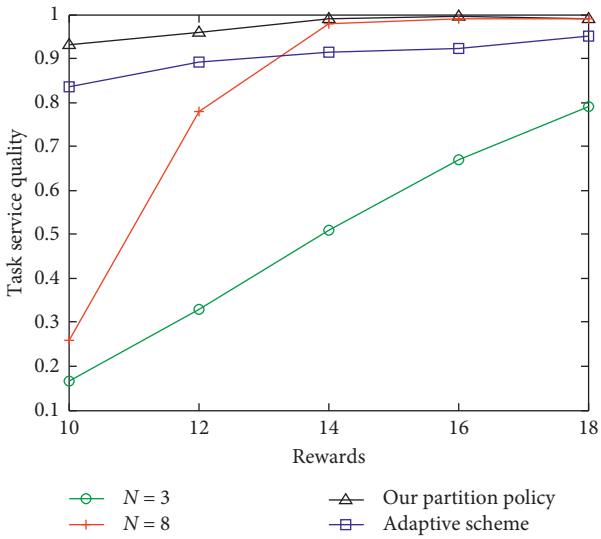

Figure 5: The task service quality varies with rewards.

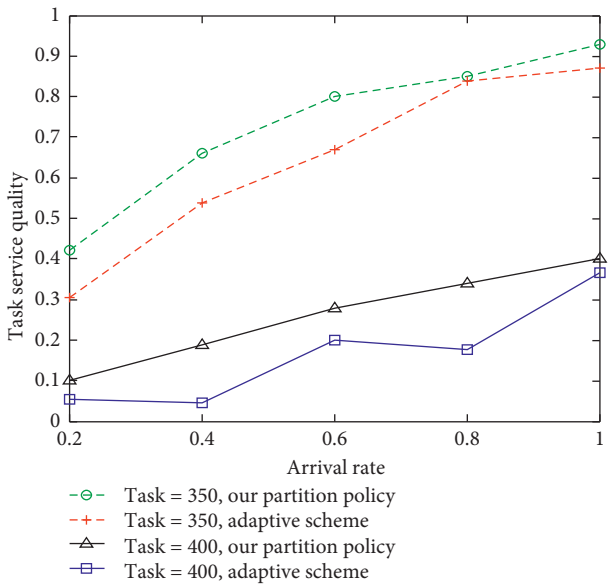

Figure 6: Task service quality varies with arrival rate.

mobile workers. The reason is that the requester will encounter more workers, and the chance for task crowdsourcing increases. When the task load Ta_i increases, the service quality decreases since it is difficult for workers with limited computing capacity to complete complex subtasks.

Figure 7 depicts the invalid number of subtask assignment attempts when the task load changes. As shown in Figure 7, our partition policy witnesses less invalid number of subtask assignment attempts than the fixed partition policy which divide the task into 5 subtasks ($N = 5$). Our partition policy is able to adjust the partition number of subtasks based on the task load and increase the accepting probability of mobile workers, while the fixed partition policy only divides task into fixed number no matter how big the task load is. Therefore, our policy realizes less invalid assignment attempts. When the

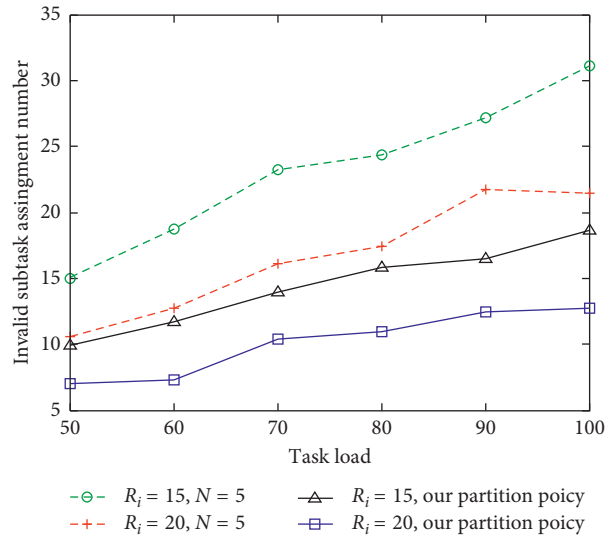

Figure 7: Invalid subtask assignment number.

reward R_i increases from 15 to 20, both our partition policy and the fixed partition policy experience less invalid number of subtask assignment attempts. The reason is that the workers are willing to accept subtasks with the high reward.

6. Conclusions

In this paper, a mobile crowdsourcing paradigm has been proposed for a task requester to obtain high-quality results. We develop a recruitment process and propose system models. Jointly considering the mobile worker features (e.g., the worker computing capacity and mobility) and task partition, a task partition problem is formulated to maximize the service quality. To solve the optimal task partition problem, a Markov chain-based solution is developed to model and perform task partition. With this policy, the requester is able to divide a complex task into optimal number of subtasks and maximize the probability that all subtasks are accepted by workers within limited time. In addition, the invalid number of subtask assignment is precisely analyzed, which is helpful to evaluate the resource consumption of requesters due to probing potential workers. Simulations show that the proposed task partition policy improves the results of task crowdsourcing.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work was supported in part by Key Research and Development Program of Shandong Province, China (no. 2017GGX10142) and in part by China Scholarship Fund.

References

[1] P. G. Ipeirotis, "Analyzing the Amazon mechanical turk marketplace," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 17, no. 2, p. 16, 2010.

[2] J. Tian, H. Zhang, D. Wu, and D. Yuan, "QoS-constrained medium access probability optimization in wireless interference-limited networks," *IEEE Transactions on Communications*, vol. 66, no. 3, pp. 1064–1077, 2018.

[3] L. Zhai, H. Wang, and C. Liu, "Distributed schemes for crowdsourcing-based sensing task assignment in cognitive radio networks," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 5017653, 2017.

[4] C. Ji, C. Zhao, S. Liu et al., "A fast shapelet selection algorithm for time series classification," *Computer Networks*, vol. 148, pp. 231–240, 2019.

[5] Y. X. Yan, L. Wu, W. Y. Xu, H. Wang, and Z. M. Liu, "Integrity audit of shared cloud data with identity tracking," *Security and Communication Networks*, vol. 2019, pp. 1–11, 2019.

[6] L. Zhai and H. Wang, "Crowdsensing task assignment based on particle swarm optimization in cognitive radio networks," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 4687974, 2017.

[7] Z. Wang, J. Xu, X. Song, and H. Zhang, "Consensus problem in multi-agent systems under delayed information," *Neurocomputing*, vol. 316, pp. 277–283, 2018.

[8] B. Hu, H. Wang, X. Yu, W. Yuan, and T. He, "Sparse network embedding for community detection and sign prediction in signed social networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 1, pp. 175–186, 2019.

[9] Q. Jiang, Y. Ma, K. Liu, and Z. Dou, "A probabilistic radio map construction scheme for crowdsourcing-based fingerprinting localization," *IEEE Sensors Journal*, vol. 16, no. 10, pp. 3764–3774, 2016.

[10] S. Jung, B. Moon, and D. Han, "Unsupervised learning for crowdsourced indoor localization in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 11, pp. 2892–2906, 2016.

[11] D. Sikeridis, B. P. Rimal, I. Papapanagiotou, and M. Devetsikiotis, "Unsupervised crowd-assisted learning enabling location-aware facilities," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4699–4713, 2018.

[12] J. An, X. Gui, Z. Wang, J. Yang, and X. He, "A crowdsourcing assignment model based on mobile crowd sensing in the Internet of things," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 358–369, 2015.

[13] H. Liu, B. Xu, D. Lu, and G. Zhang, "A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm," *Applied Soft Computing*, vol. 68, pp. 360–376, 2018.

[14] X. Deng, Y. Zheng, Y. Xu, X. Xi, N. Li, and Y. Yin, "Graph cut based automatic aorta segmentation with an adaptive smoothness constraint in 3D abdominal CT images," *Neurocomputing*, vol. 310, pp. 46–58, 2018.

[15] J. Lian, S. Hou, X. Sui, F. Xu, and Y. Zheng, "Deblurring retinal optical coherence tomography via a convolutional neural network with anisotropic and double convolution layer," *IET Computer Vision*, vol. 12, no. 6, pp. 900–907, 2018.

[16] D. C. Brabham, *Crowdsourcing*, MIT Press, Cambridge, MA, USA, 2013.

[17] J. J. Horton and L. B. Chilton, "The labor economics of paid crowdsourcing," in *Proceedings of the ACM 11th ACM Conference on Electronic Commerce*, pp. 209–218, Cambridge, MA, USA, June 2010.

[18] M. Karaliopoulos and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks," in *Proceedings of the IEEE Infocom*, Hong Kong, China, April 2015.

[19] L. Gao and J. Huang, "Providing long-term participation incentive in participatory sensing," in *Proceedings of the IEEE Infocom*, 2015.

[20] A. Kittur, E. H. Chi, and B. Suh, "Crowdsourcing user studies with Mechanical Turk," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 453–456, Hong Kong, China, April 2008.

[21] N. Eagle, "txteagle: mobile crowdsourcing," *International Conference on Internationalization, Design and Global Development*, Lecture Notes in Computer Science, vol. 5623, pp. 447–456, 2009.

[22] G. S. Tuncay and A. Helmy, "Participant recruitment and data collection framework for opportunistic sensing: a comparative analysis," in *Proceedings of the ACM MobiCom Chants*, Miami, FL, USA, September 2013.

[23] W. Chang and J. Wu, "Progressive or conservative: rationally allocate cooperative work in mobile social networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 7, pp. 2020–2035, 2015.

[24] X. Gong, X. Chen, J. Zhang, and H. V. Poor, "Exploiting social trust assisted reciprocity (star) toward utility-optimal socially-aware crowdsensing," *IEEE Transactions on Signal and Information Processing Over Networks*, vol. 1, no. 3, pp. 195–208, 2015.

[25] L. Pu, X. Chen, J. Xu, and X. Fu, "Crowdlet: optimal worker recruitment for self-organized mobile crowdsourcing," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications*, San Francisco, CA, USA, April 2016.

[26] Y. Han, T. Luo, D. Li, and H. Wu, "Competition-based participant recruitment for delay-sensitive crowdsourcing applications in D2D networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 12, pp. 2987–2999, 2016.

[27] Y. Tong, L. Chen, Z. Zhou, H. V. Jagadish, L. Shou, and W. Lv, "SLADE: a smart large-scale task decomposer in crowdsourcing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 8, pp. 1588–1601, 2018.

[28] H. Wu, H. Zhang, L. Cui, and X. Wang, "CEPTM: a cross-edge model for diverse personalization service and topic migration in MEC," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 8056195, 2018.

[29] X. Yu, H. Wang, X. Zheng, and Y. Wang, "Effective algorithms for vertical mining probabilistic frequent patterns in uncertain mobile environments," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 23, no. 3/4, pp. 137–151, 2016.

[30] M. Sharifi, S. Kafaie, and O. Kashefi, "A survey and taxonomy of cyber foraging of mobile devices," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1232–1243, 2012.

[31] U. Gadiraju, "Human beyond the machine: challenges and opportunities of microtask crowdsourcing," *IEEE Intelligent Systems*, vol. 30, no. 4, pp. 81–85, 2015.

[32] A. Singla and A. Krause, "Truthful incentives in crowdsourcing tasks using regret minimization mechanisms," in *Proceedings of the 22nd International Conference on World Wide Web-WWW'13*, Rio de Janeiro, Brazil, May 2013.

[33] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[34] J. Wu and M. Xiao, "Homing spread: community home-based multicopy routing in mobile social networks," in *Proceedings of the IEEE Conference on Computer Communications*, Turin, Italy, April 2013.

[35] A. Picu and T. Spyropoulos, "DTN-meteo: forecasting the performance of DTN protocols under heterogeneous mobility," *IEEE/ACM Transactions on Networking*, vol. 23, no. 2, pp. 587–602, 2015.