Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6643229, 15 pages
https://doi.org/10.1155/2021/6643229

Research Article

A Semiopportunistic Task Allocation Framework for Mobile Crowdsensing with Deep Learning

Zhenzhen Xie,[1] **Liang Hu**,[1] **Yan Huang**,[2] **and Junjie Pang**[3]

[1]*College of Computer Science and Technology, Jilin University, Changchun, Jilin 130012, China*
[2]*College of Computing and Software Engineering at Kennesaw State University, Atlanta, GA, USA*
[3]*Business School and the College of Computer Science and Software Engineering, Qingdao University, Qingdao, Shandong 266000, China*

Correspondence should be addressed to Yan Huang; yhuang24@kennesaw.edu

Received 13 October 2020; Revised 22 December 2020; Accepted 1 February 2021; Published 16 February 2021

Academic Editor: Xiao Zhang

The IoT era observes the increasing demand for data to support various applications and services. The Mobile Crowdsensing (MCS) system then emerged. By utilizing the hybrid intelligence of humans and sensors, it is significantly beneficial to keep collecting high-quality sensing data for all kinds of IoT applications, such as environmental monitoring, intelligent healthcare services, and traffic management. However, the service quality of MCS systems relies on a dedicated designed task allocation framework, which needs to consider the participant resource bottleneck and system utility at the same time. Recent studies tend to use a different solution to solve the two challenges. The incentive mechanism is for resolving the participant shortage problem, and task assignment methods are studied to find the best match of participants and system utility goal of MCS. Thus, existing task allocation frameworks fail to consider the participant's expectations deeply. We propose a semiopportunistic concept-based solution to overcome this issue. Similar to the "shared mobility" concept, our proposed task allocation framework can offer the participants routing advice without disturbing their original travel plan. The participant can accomplish the sensing request on his route. We further consider the system constraints to determine a subgroup of participants that can obtain the utility optimization goal. Specifically, we use the Graph Attention Network (GAT) to produce the target sensing area's virtual representation and provide the participant with a payoff-maximized route. Such a method makes our solution adapt to most of MCS scenarios' conditions instead of using fixed system settings. Then, a reinforcement learning- (RL-) based task assignment is adopted, which can help the MCS system towards better performance improvements while support different utility functions. The simulation results on various conditions demonstrate the superior performance of the proposed solution.

1. Introduction

The Mobile Crowdsensing (MCS) paradigm, as a crucial part of the current IoT ecosystem, is offering an integrated sensing capability for various aspects of the human environment, such as urban environmental monitoring [1], traffic management [2], and intelligent healthcare applications [3]. Taking advantage of the essential features of "crowdsourcing-based sensing," existing IoT applications and platforms have significantly relieved sensor resources' poverty by extending the sensing range and various sensing capabilities to support more types of sensing tasks [4]. Compared to other sensing data collection methods, we observe a significant difference between traditional sensing service and MCS systems: rather than deploying specified sensors for IoT applications with expensive cost, MCS leverages the sensing capability of multiple types of mobile devices to build a "human-in-the-loop" system [5]. In this system, each participant could be a potential worker to accomplish the sensing tasks. When they accept the task request, the participants would follow the task guidance and use the personal devices (like phones, vehicles, or smart home devices) as the sensors to collect data and then upload them to the MCS platform for further processing.

To implement this working process, MCS platforms need to select participants and provide effective rewards to make participants keep high-quality contributions. The task

allocation problem is thus essential for encouraging people to participate proactively in MCS tasks, while scheduling the sensing resource and requests properly [6]. There are two primary challenges that need to be solved for the MCS task assignment problem: the first challenge is how to accumulate and maintain a sufficient large participant pool, which refers to the participant resource bottleneck [7]; the second one is how to maximize the MCS platform benefits or satisfy task requirements under limited system budget constraints, which refers to the budget-utility contradiction [8].

In coping with the above challenges, most of the existing works assume the MCS systems already have an accurate trajectory prediction method based on sufficient mobility data and valid incentive settings. Then, the "opportunistic mode" or "participatory mode" is adopted as the primary method to organize the participants and distribute the tasks. In the opportunistic mode, the MCS platforms manage the sensing request, predict each potential worker's future trajectory, and finally distribute the sensing request to the workers who have similar routes. Thus, the workers can keep their daily routine since the sensing tasks would not disturb their original plan. Alternatively, the participatory mode means that the MCS platform needs to schedule the participants and task requests; meanwhile, the participants should follow the route to accomplish the sensing request. When they accept the sensing task, they might change their original daily routine or trip plan to follow the planned route and collect sensing data.

However, both of the two modes have several limitations. The opportunistic mode implies an essentially passive framework; the performance of the MCS platform is bounded with the conditions of participants' historical mobility and trajectory prediction method [9]. Furthermore, the MCS platform needs enough time to accumulate participant trajectory information to keep the prediction method working steadily. Furthermore, the participants' expected payoff in this mode could be high since they need to expose more historical trajectories to help the MCS platform understand their daily routines for better task scheduling, which may significantly raise privacy concerns [10]. Differently, the participatory mode uses an active way to schedule the participants and requests, but it may fail to satisfy the individual expectation of participants, especially when they are not prepared to change their original travel plan. Besides, it is notable that although the opportunistic mode considers not disturbing the participants' daily routine, they fail to provide a route to maximize the participants' payoff. As we know, unsatisfactory income can broadly impact the motivations to keep uploading high-quality data [11].

We also observe that the above limitations could be exacerbated when MCS platforms exhibit increasing sensing requests that uncertainly occur. For example, urban emergencies in healthcare and traffic scheduling [12] often need information for some targeted areas with no forecasts. These timely sensing requests come in randomly with a very short life cycle, and there is no explicit recurring need for collecting data. The sensing resource can hardly be prepared well in advance so that they enlarge the above contradiction between the increasing demand of sensing requests and the limited participant resources.

To fill the research gap, we expect to use an effortless but payoff-maximized route to motivate the participant to accomplish sensing requests. Thus, in this work, we propose a novel semiopportunistic task allocation design. Different from the existing solution, we allow participants to send the travel plan to the MCS platform, including the start and end addresses and also time constraints. Our proposed method would actively provide routing advice to maximize each participant's payoff. Compared to the task assignment adopting the opportunistic or participatory mode, our proposed semiopportunistic task allocation concept has a similar property like payoff guarantee but less impact on the participants' original travel plan. For clarity, we give an example scenario illustrated in Figure 1, and we assume that the MCS platform has seven sensing requests and five potential participants with a fixed starting point and ending point. Then, our proposed framework could provide routing advice with a maximized payoff and finally select three participants as the workers to achieve the MCS platform's performance goal.

Therefore, our work contains two parts:

(1) *Semiopportunistic sensing-based participant profiling* is aimed at resolving the participant resource bottleneck; we introduce the novel semiopportunistic sensing concept into our solution, which could be regarded as the combination of the opportunistic mode and participatory mode. This semiopportunistic sensing concept is inspired by the "shared mobility" idea [13]. The popularity of "shared mobility" gives us a novel perspective to rethink the MCS participant's bottleneck challenge. We investigated that pilot studies on real-world applications prove that "shared mobility" offers several benefits for our environmental concerns like reduced emissions and traffic congestion. Meanwhile, participants of such applications can share costs of travel or earn additional income by accepting several route changes [14, 15]

Motivated by these successful attempts, our design is aimed at adopting a similar perspective to accumulate potential participators and promote their willingness to accomplish the tasks. Specifically, unlike the opportunistic sensing method using workers to complete tasks unintentionally, we plan the participant's trip with more dedicated sensing request selection: our proposed method could insert the sensing request to their trips with no harm to their origin-destination stations and consider the individual time constraints but ensure the planned routes with a maximized accumulated payoff.

(2) *Reinforcement learning-based participant selection* is aimed at considering the system budget constraints and maximizing the utility of MCS platforms; we need to decide on a subgroup of participants to optimize the system utility under different settings. Furthermore, we expect the solution to automatically adapt to large-scale participants and choose the optimal mapping between tasks and participants under

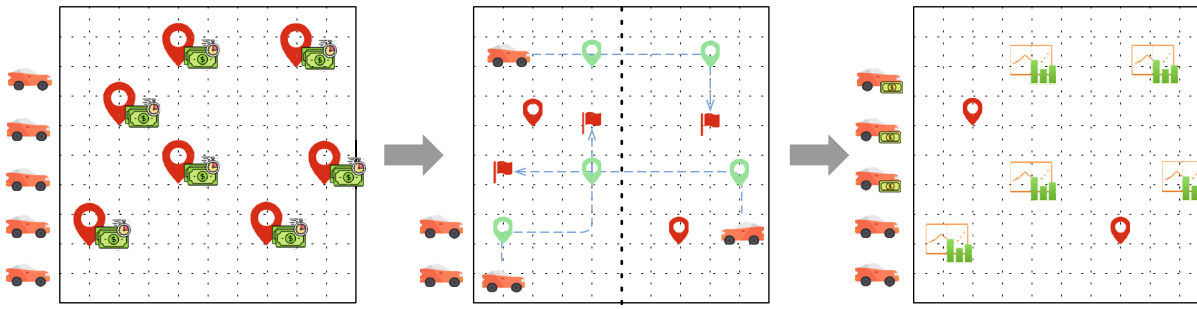

Figure 1: Framework overview.

different system constraints. Thus, we utilize the Double Deep Q-Network (DDQN) algorithm from the reinforcement learning theory to achieve an improved trade-off between stability and reactivity. It utilizes a goal-directed learning method to automatically gather the information about current participant profiles, system budget constraints, and system utility status and investigate a task allocation strategy by searching a group of participants that can guarantee the optimization goal

The fundamental differences from existing research provide the main contribution of this paper.

First, to adapt to individual participant willingness, we draw inspiration from the literature on the shared mobility concept to generate a quasi-optimal route for each worker to guarantee the maximum accumulated payoff on their daily trips.

Second, we employ Graph Attention Network techniques from graph representation learning so that the MCS platform can provide routing advice based on an improved understanding of the target sensing area. The separation of participant profiling and participant selection in our proposed framework is more tractable since the GAT method can simultaneously fuse the sensing request representations and the sensing area topology. Also, it can provide high-quality input of the participant selection method to simplify the searching process.

Finally, a participant selection strategy that utilizes the DDQN algorithm from the reinforcement learning theory is proposed to achieve an improved trade-off between stability and reactivity. It is worth mentioning the reward settings in DDQN-based participant selection decoupling the system utility goal of MCS platforms and the selection strategy, which suggests our participant selection method can adapt to different requirements for various MCS platforms.

The rest of the paper is organized as follows. Section 2 introduces related works. The system model of our proposed task allocation framework is presented in Section 3. Section 4 explains the implementation details. The simulations and results are analyzed in Section 5. Finally, the conclusion is presented in Section 6.

2. Related Works

Due to the rapid growth of big IoT data [16], the concept of MCS has been proposed to facilitate innovation in IoT sensing solutions. In various IoT scenarios like air pollution monitoring systems, urban traffic control, and noise monitoring, the MCS system offers these applications various types of sensing capability by a novel combination between humans and mobile devices. Furthermore, it can also provide plenty of information for various virtual services like information inferences [17], map services [18], and location-based social networks [19]. Task allocation becomes the major challenge to support such an MCS system since it directly affects the task response rate, sensing data quality, and economic benefit [20, 21]. This section reviews this area's recent progress and gives a brief introduction to several techniques related to our work.

2.1. Task Allocation Problem in Mobile Crowdsensing. The task allocation is an essential part of managing sensing requests and scheduling the sensing resource in MCS systems. A typical task allocation process has three primary steps: first, it characterizes current task requests under a limited system budget; second, the worker's mobility profiling with trajectory prediction or route is utilized to obtain the task acceptance rate; and finally, the platform selects a subset from existing workers to meet the system utility maximization goal like system budget, coverage ratio, or time cost of task execution.

Besides, to respond to the sensing request efficiently, existing task allocation schemes are bounded with several performance objectives, such as response latency, sensing coverage, and sensing quality of collected data. Thus, for implementing the task allocation scheme properly, recent research regards the following two problems as the primary challenges: first, select a group of participants with the maximum sensing coverage while still under the total system budget; second, translate the sensing tasks' needs into a system utility optimization goal precisely. For the first question, Zhang et al. [22] propose a coverage-oriented participant selection method and implement it as a searching process to reach the spatial coverage goal. For the latter, there are a lot of works that adopt similar thoughts with different approaches, like using enhanced greedy algorithms [7] or Genetic Algorithms (GA) [23]. Furthermore, there are also several works that consider the security concerns like sensitive information inference attacks and privacy leakage risks in the MCS task assignment [24]. For the second question, we observe some research studies are with different settings of the system utility goal. For example, Xiong et al. [25]

propose an iCrowd task allocation framework with a k-depth utility goal using a more individualized way to capture the coverage ratio of different tasks. They consider every task's spatial-temporal coverage needs and calculate the coverage by a flexible K threshold. Under such settings, the sensing resource will not be wasted on sensing tasks that cannot have enough participants.

However, we observe that most of the existing works are platform-oriented, which indicates that they consider more the platform's utility rather than the expectations of participants [26]. As a complementary, several incentive mechanisms are proposed to resolve the participant resource bottleneck by exploring multiple methods to motivate them to upload high-quality data with reasonable payoff settings [27–29], using the social network propagation theory [30] to support crowdsensing, so that they can investigate the preferences of participants to ensure the task acceptance rate. Besides, utilizing the enhanced privacy method to maintain the participant tool is another promising way, since more and more participants have privacy concerns when uploading data for IoT applications [31]. By using the influence propagation method, considering the privacy concerns of potential participants, or utilizing the recommender system-like concepts, such task assignment strategy can focus more on individual requirements to accumulate qualified workers actively [32]. There are also other types of works that consider designing a more secure MCS platform to motivate more workers to accomplish the tasks like using the blockchain-based framework [33]. Obviously, these works shift the focus from platform-oriented performance optimization to solving the trade-off between individual intentions and platform requirements.

It still faces several challenges in practical conditions; for example, the influence propagation-based task assignment needs to investigate participants' preference first, which is an additional cost that most of the applications are often unwilling to afford. What is more, these methods need much more private information than traditional approaches to meet the requirement to select the valid seed or construct the social graph.

To overcome these limitations, recent works use the Sparse Mobile Crowdsensing concept [34] instead to resolve the contradiction between the limited participant resource and the increasing need for sensing data volume. Such works use fewer participants as seed workers to collect raw data and then use data inference techniques to generate supplemental data. However, when several participants cannot provide sufficient sensing data, the sensing quality will be significantly affected.

Unlike the prior works, we propose a novel task allocation approach that uses semiopportunistic sensing to motivate potential participants to join the sensing task. The proposed method using the "shared mobility" idea reduces the extra expense for participants and provides routing advice that can further fulfill the sensing task requirements.

2.2. Graph Attention Networks. Representation learning for graph structure data is an emerging topic, and several novel neural networks are proposed to obtain graph embedding with low-dimensional representations of nodes. The Graph Attention Network (GAT) is recently regarded as a useful graph convolutional network architecture, which leverages the graph structure by the attention mechanism to extract intermediate feature representations for the nodes in the graph [35]. The sole layer of GAT is the graph attention layer, which accepts the input of a set of node features and outputs the node embedding results by performing self-attention computation between the target node and its first-order neighbors. Then, GAT uses the normalized attention coefficients to compute a linear combination of the neighbors' features and generate the final output embedding results [36]. Additionally, multihead attention is introduced in the self-attention learning process for stability consideration. By adopting a graph convolutional layer based on the masked attention mechanism, GAT satisfies several distinguished properties such as computational efficiency, considering each neighbor node's different importance, and better applicability to inductive learning problems. The interest in applying GAT to graph embedding has dramatically increased, including the traffic prediction [37], recommender system [38], and compliment techniques for knowledge graph completion [39]. Thus, we consider adopting GAT in the participant profiling step to generate routing advice for potential participants concerning different sensing area topology scales robust to the topology changes.

Reinforcement learning (RL) [40] has been used for a variety of learning tasks, ranging from resource allocation in IoT application scenarios [41], game AI like AlphaGo [42], and combinatorial optimization problems [43]. In typical settings of a model-based RL, the RL agent achieves the environment exploration and obtains an optimal policy to interact with the environment to maximize its benefits. Such a process is referred to as the Markov Decision Process (MDP), which explains an RL agent's life cycle by state, action, and reward. With the further support of deep learning, RL has proven its effectiveness on the problems requiring discrete stochastic control or continuous control with a significantly large sample size. It also shows much better performance than heuristic-based approaches. Multiple successful research attempts show that RL makes the system learn to manage the functions or resources by self-cognitive capability, which motivates us to design a feasible solution using RL to resolve the MCS task allocation problem. To the best of our knowledge, we are among the first to leverage the RL approach for enabling semiopportunistic sensing in MCS. With RL support, our solution is aimed at providing a practical self-management approach that can be easily extended to other task allocation problems with different system utility settings.

3. Problem Analysis and Formulation

This section gives the preliminary knowledge of typical task allocation problems in MCS, introduces our proposed semi-opportunistic concept as a novel solution for this problem, and explains the necessity and advantages. Furthermore, we give the general system inputs, the assumptions on

participants and the MCS platform, and the system optimization goal to explain in detail the problem formulation.

3.1. Preliminary: Task Allocation Problem in MCS. For most application scenarios, the MCS platform expects enough participant resources to improve the system utility. However, the participants always tend to search for a method that can maximize individual benefits without extra expense. Obviously, the MCS platform's participant requirements and the expected payoff of individuals lead to a contradiction for accomplishing the sensing request efficiently. Unfortunately, such circumstances could be even worse: in practical conditions, the MCS platform with only a limited system budget still needs to respond to the sensing request immediately.

There are two task allocation strategies adopted widely in current works:

(1) Opportunistic Mode. The *opportunistic mode* means that the MCS platform uses a mobility prediction method for participant selection and distributes the sensing requests to the participants with a similar trajectory. Based on the potential participant's historical trajectory data, the MCS platform can adopt machine learning models to predict each participant's daily routine, possible activity, or destinations. After that, the task assigner matches the sensing request with the participant's future trajectory and decides a number of participants as selected workers. The platform that utilizes the *opportunistic mode* for participant profiling can positively affect the delay-unaware sensing requests, the requests with a detailed time schedule, or the recurring sensing requests. In such conditions, the MCS platform can have a longer time to accumulate the participant resource, schedule the sensing request, and respond to the requests properly. While the *opportunistic mode* works efficiently for these sensing requests, it still faces several challenges: first, maintaining a large potential participant pool and accomplishing the profiling task precisely highly rely on the historical data and prediction method; second, for the MCS platform with bursts of sensing requests that occur suddenly, the *opportunistic mode* fails to recruit workers immediately since it needs more time to predict the future trajectory to prepare the participant pool.

(2) Participatory Mode. Instead of utilizing participants' possible trajectory, the selected workers should arrive at the specific locations on time to accomplish the sensing request, which means that they may change their travel plan or daily routine. The participant profiling step can then be regarded as a route planning problem with no historical trajectory requirement. Unfortunately, it also increases the sensing budget since participants may think these changes should be paid more. Besides, this mode could be unreliable, especially when the participants are unwilling to deviate from their original routines or travel plans.

Thus, we expect a task allocation framework that combines the advantages of the existing modes, while having better performance to deal with both the delay-unaware and urgent sensing requests.

3.2. A Semiopportunistic Task Allocation Concept for MCS. Many IoT applications need to make quick reactions to urgent cases, for example, the sensing request of collecting information about a traffic accident or emergency medical care activity. In such circumstances, the sensing requests occur suddenly with a fixed life cycle. The MCS platform can only have very limited time to recruit participants and schedule their sensing resources. Consider the urgent sensing requests' needs and the existing task allocation method's limitations; we proposed our *semiopportunistic mode* by utilizing the "shared mobility" concept. The standard shared mobility service means the shared use of vehicles. For example, a carpooling platform enables shared rides between drivers and passengers with similar origin-destination pairings. The recent success of shared mobility applications indicates that people tend to obtain benefits by making a few changes as possible. Thus, we are inspired to extend this concept into the MCS task allocation framework, which is referred to as the "semiopportunistic" concept in this paper. Under our concept, all potential participants can upload their travel plans with an explicit start point, destination, and deadline based on individual conditions like using carpooling applications, while the MCS platform would regard the travel plans of participants as the fixed constraints. The MCS platform then checks the current sensing requests and provides the participants with routing advice to match their travel plans while maximizing profits. Besides, we consider the system limitations of MCS (like the limited budget or different coverage requirements of task requests) and determine the final task allocation plan.

3.3. Assumptions. In our settings, we consider an MCS system with total budget B, which has a set of sensing requests $\mathrm{TR} = \{t_1, t_2, \cdots, t_n\}$ that occur randomly in a target sensing area L and a set of potential participants $W = \{w_1, w_2, \cdots, w_k\}$ posting their travel plans, while waiting to be paired with the maximized profit path planning advice. We assume target sensing area L is composed of a set of cells, and each cell refers to a possible sensing location. A task request t_i has a target location loc_i, a minimal sensing coverage threshold q_i, a task deadline time_i, and a specific incentive val_i in terms of credits or monetary rewards to encourage participants to respond to this sensing request. Note that the value of q is the minimum coverage required of this sensing request to characterize the target region. When there is less than q_i participants who accept the individual task request, the sensing coverage of t_i will be set to zero. Besides, q of each task request could be different to adapt to each task's needs.

Then, we define a task request as follows:

$$t_i = \langle \mathrm{loc}_i, \mathrm{val}_i, \mathrm{time}_i, q_i \rangle, \quad t_i \in \mathrm{TR}. \tag{1}$$

Furthermore, the travel plan of each potential participant is denoted by a fixed start point sta, a destination des, and also a time constraint time, which indicates that the participant must arrive at the destination location no later than

time. Then, the travel plan can be defined as follows:

$$w_j = \langle \text{sta}_j, \text{des}_j, \text{time}_j \rangle, \quad w_j \in W. \tag{2}$$

The routing advice for participant w_j can be defined as follows:

$$p_{w_j} = \left\{ \text{sta}_j, t_{w_j^1}, \cdots, t_{w_j^m}, \text{des}_j \right\}, \quad p_{w_j} \in P. \tag{3}$$

After the MCS platform provides the routing advice for each potential participant, it checks the system budget and selects a subset of participants W' as the selected workers that could maximize the system utility. Here, we define the system utility as follows:

$$u_{t_i}^{W} = \min \left\{ \left\lfloor \sum_{w_j \in W'} \frac{I_{w_j}^{t_i}}{q_i} \right\rfloor, 1 \right\}, \tag{4}$$

where

$$I_{w_j}^{t_i} = \begin{cases} 0, & \text{when } t_i \text{ not in } p_{w_j}, \\ 1, & \text{when } t_i \text{in } p_{w_j}. \end{cases} \tag{5}$$

Finally, the selected participants would accept the routing advice with a group of sensing task requests inserted in their original trip plan and then sequentially visit the location of each task to collect sensing data.

3.4. Problem Definition. In our proposed task allocation framework based on the semiopportunistic concept, our primary objective is to select a subset of participants with payoff-maximized routes while maximizing MCS platforms' system utility.

Specifically, with the constraint of the given travel plan of the participant, our MCS platform tends to provide the participant w_j with the routing advice p_{w_j} having maximized payoff(p_{w_j}), which can be denoted as follows:

$$p_{w_j}^* = \underset{p_{w_j}}{\text{argmax}} \sum_{t_i \in p_{w_j}} \text{val}_{t_i}. \tag{6}$$

The objective function of system utility is further defined as follows:

$$\begin{aligned} &\max \sum_{t_i \in T} u_{t_i}^{W} \\ &\text{s.t.} \sum_{w_j \in W'} \text{payoff}\left(p_{w_j}\right) \le B, \end{aligned} \tag{7}$$

We can find it is a multiobjective optimization problem when we expect to maximize the above two objectives simultaneously. Unlike the current solutions using Pareto optimality to resolve this problem of having scale limitations, we implement our objective problem in two parts: firstly, for every potential participant, we calculate a route with the maximized payoff to accomplish the participant profiling stage; we further select valid participants from the participant-route set to find the subset with maximized system utility under a limited system budget.

4. Implementation: MCS Task Allocation Framework with Deep Learning Support

This section gives the detailed implementation of our proposed semiopportunistic concept for the MCS task allocation problem, which includes two parts: the participant profiling and the participant selection. First, we give the system overview of our proposed framework. Second, we propose the participant trajectory profiling method based on GNN techniques. We give the detailed GNN model structure and the primary training process to demonstrate the detailed implementation. Finally, an RL-supported participant selection algorithm is proposed. We elaborate on the MDP underlying our method and further explain it by a simplified example.

4.1. Framework Overview. Our proposed working process of the MCS task allocation has three primary stages of fulfilling various types of task requests efficiently, which includes: task request initialization, trajectory profiling for potential participants, and participant selection.

4.1.1. Task Request Initialization. This component accepts the sensing request that randomly occurs. Each request is tagged with a list of the necessary information, including the target location, the maximum requirement of participants, the time of the deadline, and incentive settings. Meanwhile, the platform allows multiple types of sensing requests at one time, which could have different sensing requirements like coverage ratio or time constraints.

4.1.2. Potential Participant Profiling. This component calculates a route with the maximum profits for each participant using the personal travel plan as the available time, start, and destination constraints. Specifically, to motivate the participants to respond to the task requests quickly, the MCS platform calculates the routing advice with the maximum payoff for each participant. Obviously, it is a critical challenge since the routing advice task can be reduced as the orienteering problem [44], which is NP-hard. That means, when the MCS platform has a large set of potential participants and sensing requests, the participant profiling could be the bottleneck. Thus, we introduce an attention-based encoder-decoder model to implement the participant profiling step. Unlike the greedy-based algorithms with approximation bound determined by a fixed utility function, our attention-based model has improved flexibility to adapt to different utility function settings and different sensing area topology scales. It can also adapt to the target sensing area topology changes to adjust to different urban environment dynamics without introducing much extra computation cost.

4.1.3. Participant Selection. Given the participant profiling information, this component determines a group of

participants that can fit the system utility goal under a fixed system budget constraint. In our proposed framework, we give a reinforcement learning-based solution for searching the participant-route pairs. In the typical RL settings, the MCS platform can serve as the RL agent to collect the information of the sensing requirements and participant profiles; then, it actively searches for an optimal subset of participant-route pairs as the selected workers to accomplish the sensing tasks.

4.2. GAT-Based Solution for Potential Participant Profiling. For the sensing requests that randomly occur but expect a quick response, the MCS platform needs to motivate each potential participant efficiently. Thus, in our proposed task allocation framework, the participant trajectory profiling stage serves as an essential part to motivate the participant by providing them with a payoff-maximized route. However, the sensing area could be large. The topology may change due to urban traffic management, accident, or abnormal climate, which indicates that we need a flexible and efficient solution to resolve the topology dynamic challenges while ensuring good scalability.

Specifically, we formalize the routing problem by using a Graph Attention Network to produce embeddings of the target sensing area and then compute the routing advice. As we know, the graph embedding representation can provide a powerful solution to adapt to large-scale urban sensing scenarios. It also supports various types of information that can be represented by the nodes' attributes [45, 46]. Thus, the topology of the target sensing area can be represented as a graph G with a node set $\boldsymbol{\pi} = \{\pi_1, \pi_2, \cdots, \pi_n\}$, where each node could be a potential sensing location. Then, the attention-based encoder-decoder model proposed in [43] is adopted to implement the participant profiling component; we illustrate the working process in Figure 2.

Encoder. The encoder takes each node π_i and its feature val_i in $\boldsymbol{\pi}$ as inputs, and when there is no sensing request at node π_i, the val_i would be 0. The initial node embedding of π_i with parameters W and b can be represented by $h_i^0 = W[\pi_i, \text{val}_i] + b$. By using the N attention layer, h_i^0 can learn the relations with all the other nodes and update itself as h_i^N. Each attention layer has two sublayers, including a multihead attention (MHA) layer and a feedforward (FF) layer. The encoder then uses all the node embedding results to produce the graph embeddings of the target sensing area as follows: $\bar{h}^N = (1/n)\sum_{i=1}^{n} h_i^N$.

Decoder. At time t, the decoder takes the node embedding h_i^N, the sensing area embedding $\bar{h}^N$, and the travel plan of the participant w_j including the start location, destination location, and time constraint $\langle \text{sta}_j, \text{des}_j, \text{time}_j \rangle$ as the inputs. Then, the decoder produces the context embedding $h_{w_j}^N$ of the participant w_j by considering the sensing area embedding $\bar{h}^N$, the location of the previously selected task request at time $t-1$, and the destination des_j, which is denoted as follows:

$$h_{w_j}^N = \begin{cases} \left[\bar{h}^N, h_{\text{sta}_j}^N, h_{\text{des}_j}^N\right], & t = 1, \\ \left[\bar{h}^N, h_{\tau_{w_j}^{t-1}}^N, h_{\text{des}_j}^N\right], & t > 1. \end{cases} \tag{8}$$

After we have $h_{w_j}^N$ and the node embedding of nodes having sensing requests, the decoder uses MHA to get a new embedding result $h_{w_j}^{N+1}$, which indicates the correlation between $h_{w_j}^N$ and other nodes with sensing requests at time t. To produce the routing advice, we use a single head attention layer to determine the next node that the participant w_j should visit. To adapt to the time constraint of w_j, we mask the nodes with a task finishing deadline that the participant w_j cannot visit them within his remaining time. We also mask the nodes that are already visited to ensure the participant would not accomplish a sensing request twice. Such a working process repeats for several iterations until the remaining time runs out. Finally, we can obtain the routing advice for each potential participant. To ensure the final route is payoff-maximized, several training methods can be used to train the encoder-decoder network like the actor-critic algorithm [47] or REINFORCE with deterministic greedy rollout baseline [48].

4.3. Participant Selection with RL Support. Before diving into the details of our proposed RL-based participant selection, we first depict the MDP that formalizes the target problem about selecting a subset of participants to optimize the system utility. Then, we use a tabular Q-learning example to describe the RL working process and further extend it by introducing DNN to ensure our method can adapt to the practical conditions of large-scale participants and sensing requests.

RL is a goal-directed learning approach that uses an interactive manner to explore the environment and investigate how an agent can derive the maximum accumulated reward. To formalize the problem space, the Markov Decision Process is adopted to describe the interactions between the RL agent and the environment, which has three essential components: state, action, and reward. The *state* includes the direct knowledge of the problem to indicate what we have already known at a specific time slice. Then, the RL agent learns how to make *actions* at each state, and it always expects to find the optimal action-state mappings (optimal policy) to maximize the cumulative *reward* as a learning result. Although the problem space may be full of uncertainty, RL can automatically explore the environment and recognize the optimal policy to help the agent always make the right decisions under different conditions. From the above facts, we observe an explicit relation between RL techniques and our participant selection problem, which can be fully explained by the following MDP (Markov Decision Process) formulation.

4.3.1. MDP Formalization. To adopt the RL approach for the participant selection problem, we regard the MCS system as the environment. The RL agent interacts with the

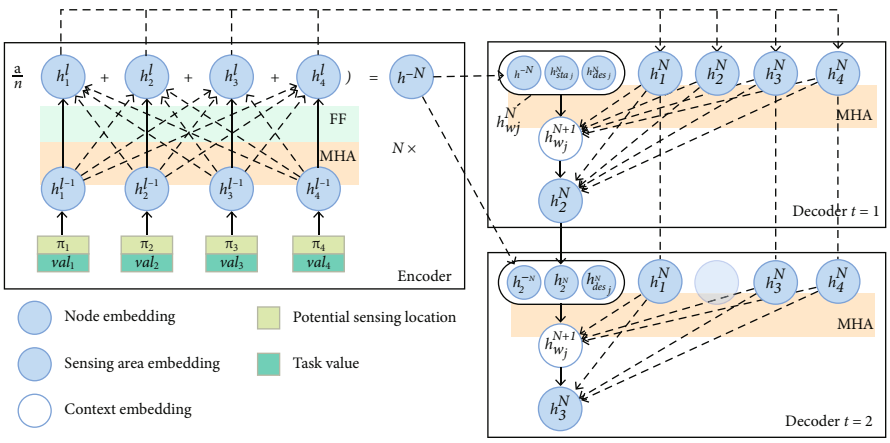

Figure 2: Model structure and working process.

environment to gather information about the available participant profiles (planned routes), current task requests, and system budget requirements. Through the RL agent's exploration and exploitation, the agent learns the optimal policy to decide the participant-task mappings to achieve better system utility. To implement the above process, we first define the state, action, and reward as three primary elements and depict our MDP in Figure 3.

State. We represent the functional conditions of the MCS platform—the current task request TR, system budget B, available participant resources, and related profiles P—as the state of our MDP. Thus, the state s_t at time t can be represented as $s_t = (\mathrm{tr}_t, b_t, p_t)$ that belongs to the state space S.

Action. We assume there are k participant profiles in the current MCS platform, and at each time step, the RL agent selects one participant to accomplish the tasks in his route. Then, the action space is given by $A = \{1, 2, \cdots, k\}$ and action a_t represents the decision on participant selection. With each available action, the RL agent observes a state transition. When a participant's route is selected to improve the system utility, the next state will be updated accordingly.

Reward. The reward signal guides the agent towards an optimal solution for the target problem. In our problem settings, the objective is maximizing the total system utility under resource constraints. Specifically, we set the reward at each time step as the system utility. Note that although the agent can receive an immediate reward for each action participant selection, the RL agent focuses more on maximizing the cumulative reward to ensure the MCS system receives the largest utility value by the selected participant group. To ensure the RL agent can be farsighted, we use the discount rate $\gamma \in [0, 1]$ as the parameter to determine the present value of the future reward. As its value approaches 1, it means that the RL agent takes future rewards into account more strongly.

4.3.2. A Tabular Q-Learning Example. For small-scale applications, the MCS system can use tabular Q-learning, a value-based reinforcement learning algorithm that uses an evaluation concept—Q-function—to derive the optimal policy. In the tabular Q-learning-based participant selection algorithm, it utilizes the Q-function denoted as $Q(s, a)$ to calculate the maximum expected future reward (system utility) that the agent will get if it takes action a at state s. Afterward, each possible state-action pair's Q-value will be stored in the Q-table $Q_{|S|\times|A|}$. Thus, the RL agent can evaluate each participant selection in terms of reward, derive the estimated value of $Q(s, a)$, and record this value in $Q_{|S|\times|A|}$. To find the optimal policy, the Q-table $Q_{|S|\times|A|}$ will be further iterated and updated by the Bellman equation with learning rate α as follows:

$$Q(s, a) = Q(s, a) + \alpha\left(r + \gamma \max_a Q\left(s', a\right) - Q(s, a)\right). \quad (9)$$

After $Q_{|S|\times|A|}$ is updated, the available action space will be changed accordingly so that the agent can select another participant satisfying the current budget constraints. Through the tabular Q-learning process, the agent follows the ε-greedy policy, that is, with $1 - \varepsilon$ possibility to select the participants with the largest Q-value until the Q-table $Q_{|S|\times|A|}$ is converged. That means, under each state s, the agent can select the participant with the largest Q-value by searching the Q-table $Q_{|S|\times|A|}$ as an optimal policy.

To further explain the training process, we introduce a simplified participant selection problem as a toy example illustrated in Figure 4.

In our example, we assume that the MCS system has 10 task requests with different participant requirements denoted as task$_i d : q$; for example, tr0 : 2 indicates that task 0 has a minimum requirement of 2 participants. We also have 4 participants with profiles (planned routes) denoted as Participant A (tr0 – tr1 – tr2), Participant B (tr1 – tr4 – tr7), Participant C (tr3 – tr6 – tr7), and Participant D (tr7 – tr8). For the tabular Q-learning process, we set the discount factor γ and learning rate α as 1. First, at state s_0, the MCS system has not chosen any participants, and the Q-table is initialized with 0. Then, the RL agent begins to interact with the environment by a random policy, and we assume it chooses action a_2 that selects *Participant B*. The RL agent receives an immediate reward in terms of current system utility, and we observe a state transition from s_0 to s_1. We can

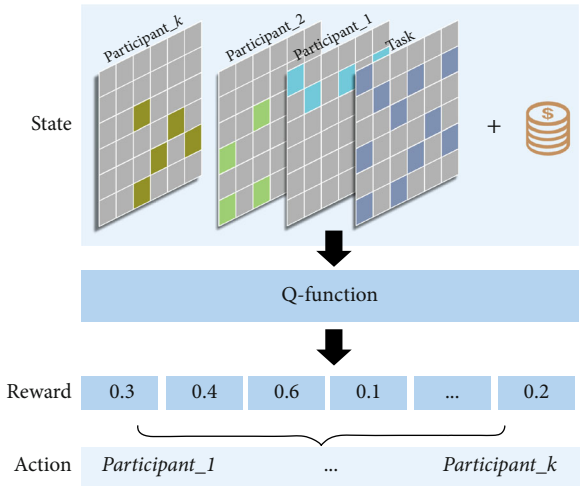

Figure 3: Illustration of the MDP.

now update the $Q(s_0, a_2)$ in the Q-table as $Q(s_0, a_2) = 0.2 + 0 = 0.2$.

Similarly, at the next time step, we use the simple greedy policy and choose action a_4 that selects *Participant D* to update $Q(s_1, a_4)$ as $Q(s_1, a_4) = 0.3 + 0 = 0.3$. After several rounds, at time step t_k, the Q-table is changed as in the figure: t_k. At this time step, we assume the RL agent goes back to s_0 and it checks the current Q-table and recognizes that under s_0, $Q(s_0, a_2)$ has the largest Q-value. Hence, the RL agent executes action a_2 and observes a state transition from s_0 to s_1. According to the Q-table at time step t_k, the largest expected future reward is 0.3. Hence, at the next time step t_{k+1}, $Q(s_0, a_2)$ will be updated as $Q(s_0, a_2) = 0.2 + 0.3 = 0.5$. That means, when the RL agent meets s_0 at future time steps, it has a higher possibility to select action a_2, since the Q-table has explicit evidence that it can lead to higher accumulating rewards. As an iterative process, the above operations repeat several times and the Q-table will be improved at each iteration so that the Q-value is approaching the practical state-action value.

4.3.3. DDQN-Based Participant Selection Algorithm. Although tabular Q-learning offers an effective solution for our simplified participant selection problem, we still require a method meeting the practical requirements, such as large-scale participants, system budget constraints, and various attributes of task requests. Thus, instead of using the Q-table in the tabular Q-learning method, we choose the Convolution Neural Network (CNN) as a Q-network to obtain the estimation of the Q-function. Specifically, we represent the state of our MDP—the current allocation task request waiting to be scheduled, available participant resource, and planned route profiles—as a $m \times n \times (i + j)$ matrix. Here, $m \times n$ indicates the target map has $m \times n$ cells, i is the number of participants, and j is the number of task request' attributes. With two convolutional layers and one fully connected layer, our proposed CNN is used to extract the above state matrix's primary feature and output the Q-function value for each state. We further illustrate the Q-network training via the DDQN-based algorithm in Algorithm 1 as follows:

In the DDQN-based participant selection algorithm, we utilize the experience replay technique to break the temporal correlations that lie in various training episodes. A replay buffer with a fixed size is utilized to mix experiences at different time steps for the Q-network updates. At the beginning of this algorithm, the Q-network is initialized to a random value (Line 3). Meanwhile, the initial state s_0 feeds the Q-network and the RL agent selects an action under the ε-greedy policy to start the first training episode (Lines 5 and 6). Next, the state transition $\{s, a, r, s'\}$ is stored in the replay buffer (Line 8). When executing an action, the algorithm checks the available system budget to ensure the remaining budget can afford the next participant selection (Line 9). Then, given the replay buffer, the agent samples a random minibatch and updates the Q-network using the following loss function (Lines 14 and 15):

$$\text{Loss}(\theta) = \frac{1}{m} \sum_{\text{minibatch}} \left[\left(r + \gamma \max_{a'} Q_{\theta'}(s', a') - Q_{\theta}(s, a) \right)^2 \right]. \tag{10}$$

Here, the target Q-network is an independent estimator that updated slower than the Q-network, to avoid maximization bias by disentangling updates from biased estimate values.

5. Performance Evaluation

This section validates our proposed method through extensive simulations of multiple application scenarios. We first introduce the experimental setup, parameter settings, and baseline algorithms. Then, we demonstrate the performance comparison result in multiple scenarios having different system budgets, numbers of participants, or numbers of sensing requests.

5.1. Dataset and Selected Parameters. For generating the trip plans of the potential participants, we adopt the T-drive dataset [49] to provide the start location and destination, which contains the GPS trajectories of 10,357 taxis from Feb 2nd to Feb 8th, 2008. We then select 1000 travel plans and randomly generate a deadline as the time constraint of each trip plan to form our potential participant pool. Furthermore, since we propose a two-stage solution to implement the semi-opportunistic MCS task assignment concept we defined in this paper, four primary factors that affect the simulation results are selected to describe the validity and performance: the total system budget of the MCS platform, the number of task requests, the value of the task, and the number of participants.

5.2. Experimental Setup and Settings. We implement our work on the PyTorch platform. The encoder-decoder model in [19] is adopted to provide the route with a maximized payoff. The minimum requirement of participants for each task request is randomly generated from $(2, 15)$, while the value

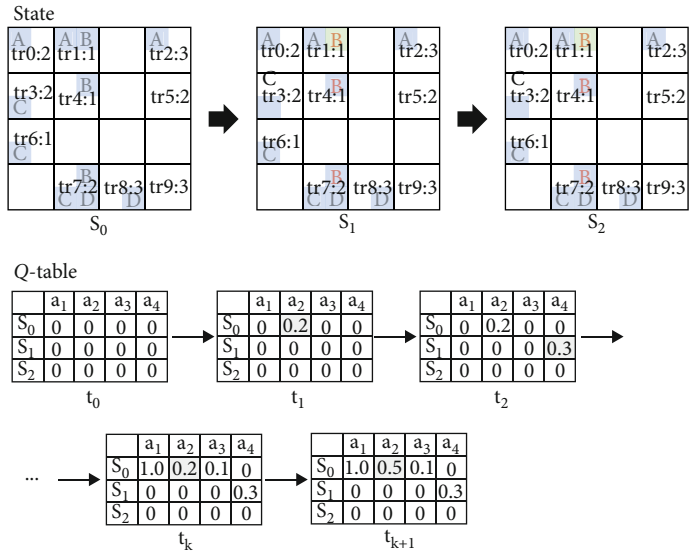

Figure 4: A tabular Q-learning example.

Participant selection based on DDQN.
1: Initialize Q-network, Q' target network with θ
2: **while** In each episode **do**
3: Initialize s
4: **for** step in episode **do**
5: With probability ε, select a random action a
6: otherwise select $a = \max_a Q_\theta(s; a)$
7: $r, s' = \text{perform\_action}(s, a)$
8: store transition $\{s, a, r, s'\}$ in replay_buffer
9: **if** current system budget cannot afford any participant **then**
10: s is terminal state
11: **else**
12: $s = s'$
13: **end if**
14: sample random minibatch(m) of transitions from replay_buffer
15: perform minibatch gradient descent
16: every updated period $T, \theta' = \theta$
17: **end for**
18: **end while**

Algorithm 1:

of the task request is randomly generated from $(10, 50)$. For clarity, we take the routing advice for 15 participants as an example shown in Figure 5. The sample routes are the example solutions for a sensing area consisting of 100×100 cells and 200 sensing requests using the GAT-supported routing method. For the proposed RL-based task allocation solution using the DDQN algorithm, the Q-network is CNN-based. The replay buffer size is 1000, and the minibatch size for sampling is 32. We set the learning rates of the Q-network as 10^{-3} and the discounting factor γ as 0.99.

5.3. Baseline Algorithm. Since no previous works have studied the task allocation method with the semiopportunistic concept via deep learning support, we select the following baseline methods to accomplish the comparative studies:

Random allocation. This method randomly selects participants from the potential participant pool until meeting the total system budget. Since the random character may affect the simulation result, in our simulation, we repeat this method for 10 times and the average overall utility is utilized as the final result.

Low-payoff first allocation. This is a single-loop greedy algorithm which tends to select more participants to obtain higher system utility. It orders the potential participants from the minimal total payoff and then selects a subgroup of participants having a route with a low total payoff until the total system budget runs out.

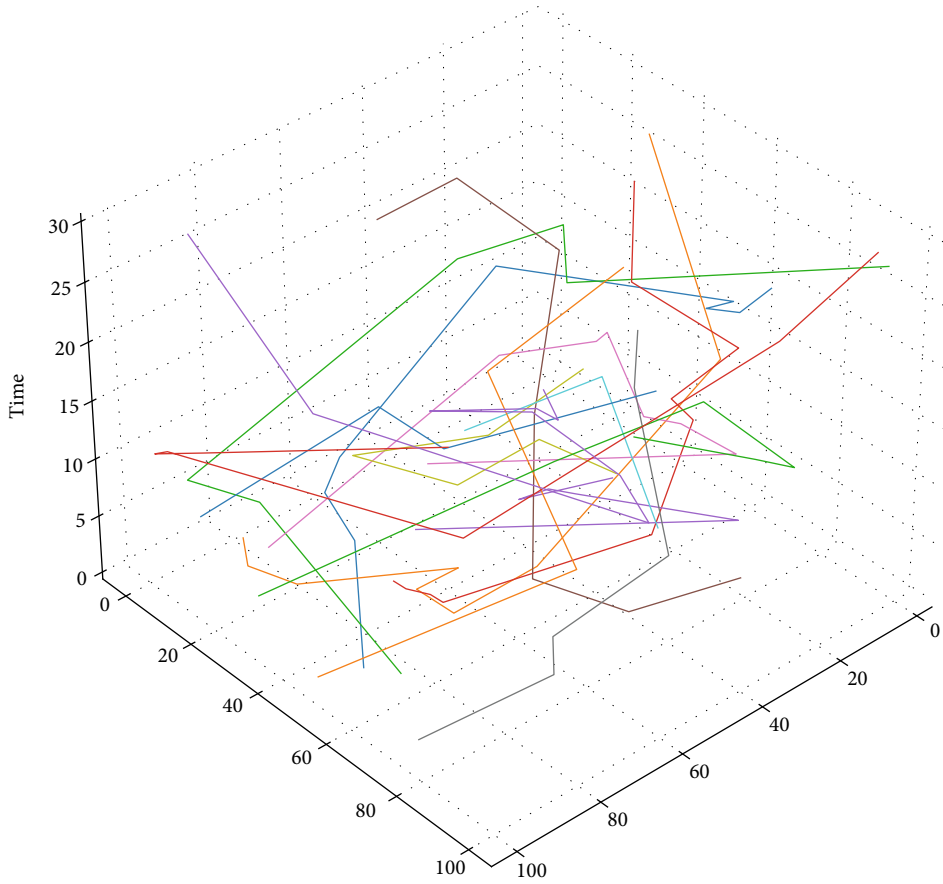

Figure 5: Example routing advice with a maximized payoff.

Long-route first allocation. This is a single-loop greedy algorithm which tends to select participants with the long-route first method to reach the system utility goal, since the longer route means this participant could obtain more sensing request than others. It orders the participant by the route length and then selects a subgroup of participants with the longer route until the total system budget runs out.

5.4. Performance Comparison via Multiple Scenarios

5.4.1. Different Numbers of Task Requests. Figure 6 depicts the performance comparison result on the system utility among the DDQN-based allocation and other baseline methods under the different numbers of task requests, where we fix the number of selected participants as 200. We observe that in the beginning, the performance difference on the system utility is not very significant due to the very small task numbers. When the system utility decreases with the increasing number of tasks for all three methods due to the limited resource's enhanced competition, our proposed algorithm outperforms other baseline methods to obtain higher utility for different settings for the number of task requests. When the number of requests increases while all the three methods' utility decreases, our method's system utility still outperforms the other methods, and the utility decreases more steadily. Although we can observe the same decreasing trend with the other three methods, they fail to obtain a similar utility performance as our method does. Such results indicate that our method works well at both the small- and large-scale sensing requests.

5.4.2. Different Values of the Total System Budget. Figure 7 gives the results for the system utility changes among all the four allocation methods when the system budget is varied. In this experiment, we define the system budget as $B = \sum_{i \in \mathrm{TR}} \mathrm{val}_i \times q_i$. When the total budget increases, it indicates that the MCS platform can have more participants to accomplish the request. Therefore, we observe an increase of all four methods. At the beginning, except for the utility of the random method that is dragging by the random character, all the other three methods have a similar increasing trend. Then, the performance gap between our method and the other three methods becomes larger. From Figure 7, we find the random method has the smallest utility increment while the low-payoff first method and the long-route first method have a very similar trend in the end. At the same time, our method is more stable and can always obtain higher utility when the system budget changes.

5.4.3. Performance Comparison for Different Values of q of Each Task Request. Figure 8 shows the comparison result when we are varying the values of q for each task request. We generate q, the minimum requirement of the participant for each request, by randomly choosing a number from (5,

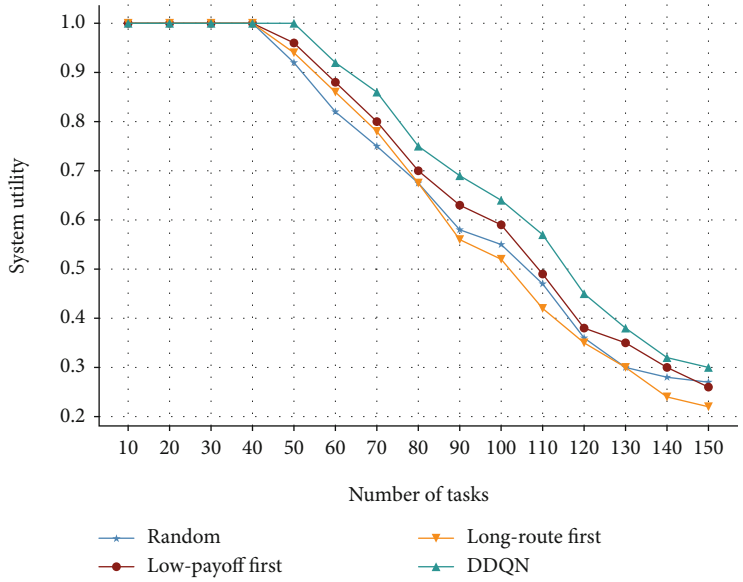

Figure 6: Performance comparison for the various numbers of tasks.

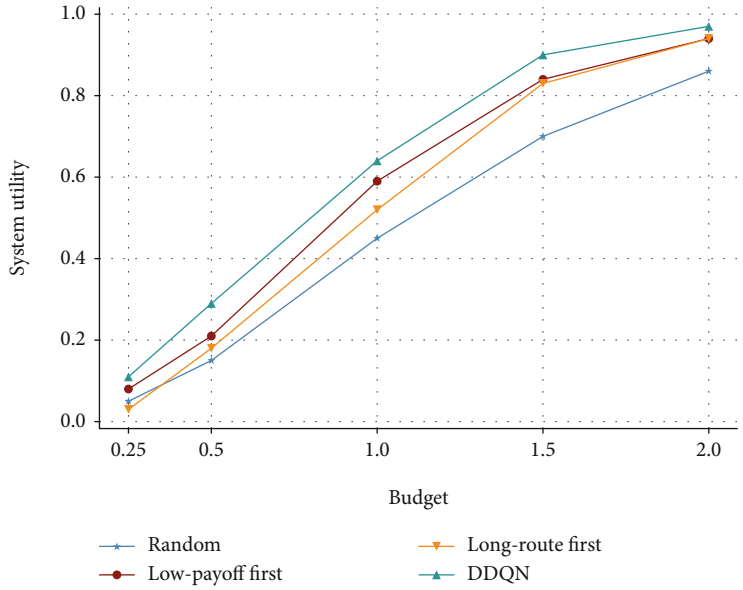

Figure 7: Performance comparison for the different values of the budget.

15). Figure 8 plots the utility changes when the value of q varies, which can simulate the multiple types of the minimum thresholds for different application scenarios. When the requirement of the task request increases, we observe that the system utility decreased since the total budget and potential participant pool are fixed. The system utility of the random method decreased sharply, while the low-payoff first method and the long-route first method represent similar trends with our method; however, when the value of q increases from 8, we observe that both of them have a noticeable decline. Thus, compared to other baseline methods, our method can obtain a significantly high system utility that decreases more steadily during the value changes of$q$.

5.4.4. Number of Assigned Participants for Each Task Request. Figure 9 represents the changes in the number of assigned participants when varying the total number of task requests. In this experiment, we assume all the 100 task requests have the same minimum threshold $q = 5$, which is represented by the dotted line in Figure 9. We can see that for each task request, the long-route first method tends to select the participant with the longer route to obtain the system utility goal; however, it could waste several participants. Meanwhile, we observe a similar result of the low-payoff first method. For example, we can notice for task id:40 that it allocates 7 more participants, which is significantly larger than the minimum threshold. Compared with the three baseline models, our

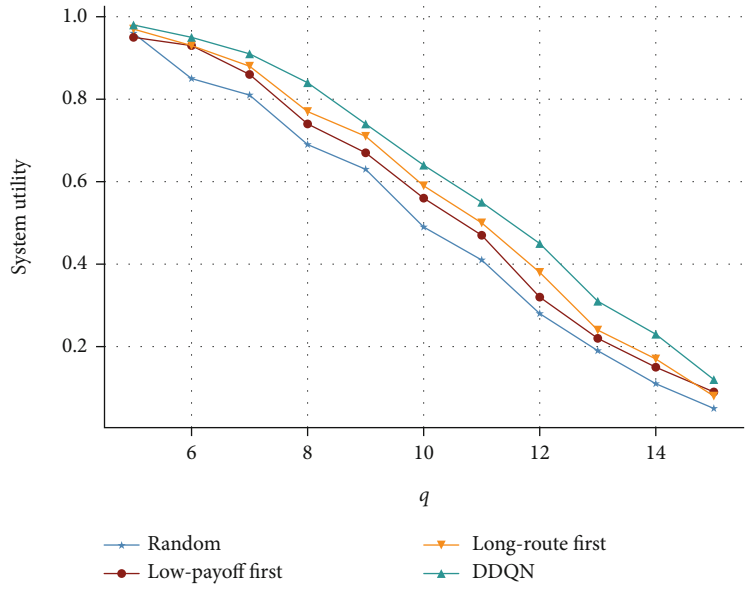

Figure 8: Different values of q for each task request.

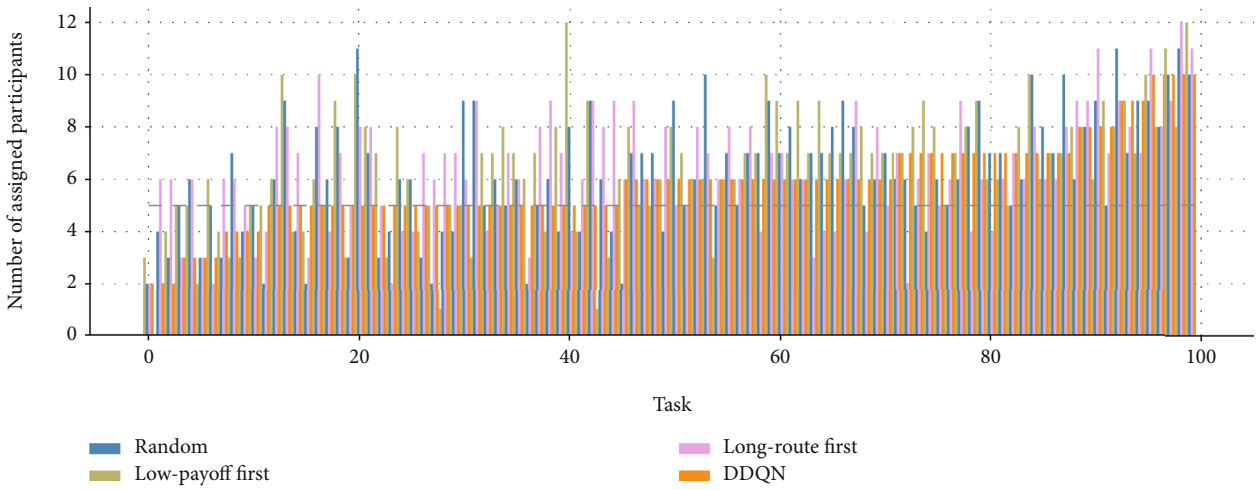

Figure 9: Number of assigned participants for each task request.

method tends to meet the threshold with less waste of participant resources so that we can see that the number of assigned participants of all the 100 task requests is closer to the dotted line.

6. Discussion and Conclusion

This paper studied the task allocation problem in MCS systems by introducing a novel semiopportunistic concept inspired by "shared mobility" applications. Meanwhile, we aim to maximize the payoff of the participant by producing a well-considered route. Then, we use the reinforcement learning technique to select a subgroup of participants under the system utility optimization goal. Our proposed solution has several advantages. First, to implement our proposed framework efficiently, we adopt a representation learning approach to produce the target sensing area embeddings. At the same time, output a payoff-maximized route for each participant. Second, the reinforcement learning-based participant selection algorithm is proposed for selecting a subgroup of participants that can meet the system utility goal. Unlike traditional solutions using greedy-based or heuristic-based algorithms, our proposed framework and its implementation can support large-scale sensing requests and various types of utility optimization goals. Finally, extensive simulations indicate our solution outperforms the baseline methods under various conditions. In the future, we plan to extend our method into the social network-based MCS platforms to investigate the participant profiling problem with social influence analysis. We expect that investigating social relationships among participants by using deep learning approaches could be a different solution to solve the participant resource bottleneck.

Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Key Research and Development Plan of China under Grant 2017YFA0604500, in part by the National Natural Science Foundation of China under Grant 61701190, in part by the Youth Science Foundation of Jilin Province of China under Grant 20180520021JH, in part by the Youth Sci-Tech Innovation Leader and the Team Project of Jilin Province of China under Grant 20170519017JH, in part by the Key Technology Innovation Cooperation Project of Government and University for the whole Industry Demonstration under Grant SXGJSF2017-4, and in part by the Program for Innovative Postdoctoral Talents in Shandong Province.

References

[1] D. Wu, T. Xiao, X. Liao et al., "When sharing economy meets IoT: towards fine-grained urban air quality monitoring through mobile crowdsensing on bike-share system," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 2, pp. 1–26, 2020.

[2] X. Wang, Z. Ning, X. Hu et al., "A city-wide real-time traffic management system: enabling crowdsensing in social internet of vehicles," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 19–25, 2018.

[3] M. Mehdi, G. Mühlmeier, K. Agrawal, R. Pryss, M. Reichert, and F. J. Hauck, "Referenceable mobile crowdsensing architecture: a healthcare use case," *Procedia Computer Science*, vol. 134, pp. 445–451, 2018.

[4] Y. Wang, Z. Cai, Z.-H. Zhan, B. Zhao, X. Tong, and L. Qi, "Walrasian equilibrium-based multiobjective optimization for task allocation in mobile crowdsourcing," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 4, pp. 1033–1046, 2020.

[5] H. Yang, F. Li, D. Yu, Y. Zou, and J. Yu, "Reliable data storage in heterogeneous wireless sensor networks by jointly optimizing routing and storage node deployment," *Tsinghua Science and Technology*, vol. 26, no. 2, pp. 230–238, 2021.

[6] Z. Duan, W. Li, X. Zheng, and Z. Cai, "Mutual-preference driven truthful auction mechanism in mobile crowdsensing," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1233–1242, Dallas, TX, USA, October 2019.

[7] S. Reddy, D. Estrin, and M. B. Srivastava, "Recruitment framework for participatory sensing data collections," in *Pervasive Computing. Pervasive 2010*, P. Floréen, A. Krüger, and M. Spasojevic, Eds., vol. 6030 of Lecture Notes in Computer Science, , pp. 138–155, Springer, 2010.

[8] Y. Liu, L. Kong, and G. Chen, "Data-oriented mobile crowdsensing: a comprehensive survey," *IEEE Communication Surveys and Tutorials*, vol. 21, no. 3, pp. 2849–2885, 2019.

[9] J. Li, H. Jiao, J. Wang, Z. Liu, and J. Wu, "Online real-time trajectory analysis based on adaptive time interval clustering algorithm," *Big Data Mining and Analytics*, vol. 3, no. 2, pp. 131–142, 2020.

[10] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial IoTs," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.

[11] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A survey on mobile crowdsensing systems: challenges, solutions, and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2419–2465, 2019.

[12] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6492–6499, 2019.

[13] R. Tachet, O. Sagarra, P. Santi et al., "Scaling law of urban ride sharing," *Scientific Reports*, vol. 7, no. 1, article 42868, 2017.

[14] G. Laporte, F. Meunier, and R. W. Calvo, "Shared mobility systems: an updated survey," *Annals of Operations Research*, vol. 271, no. 1, pp. 105–126, 2018.

[15] N. Chan and S. Shaheen, "Ridesharing in North America: past, present, and future," *Transport Reviews*, vol. 32, pp. 112–193, 2012.

[16] Z. Cai and Z. He, "Trading private range counting over big IoT data," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 144–153, Dallas, TX, USA, July 2019.

[17] Y. Liang, Z. Cai, J. Yu, Q. Han, and Y. Li, "Deep learning based inference of private information using embedded sensors in smart devices," *IEEE Network*, vol. 32, no. 4, pp. 8–14, 2018.

[18] X. Fan, X. He, C. Xiang et al., "Towards system implementation and data analysis for crowdsensing based outdoor RSS maps," *IEEE Access*, vol. 6, pp. 47535–47545, 2018.

[19] J. Li, Z. Cai, M. Yan, and Y. Li, "Using crowdsourced data in location-based social networks to explore influence maximization," in *35th Annual IEEE International Conference on Computer Communications, INFOCOM*, pp. 1–9, San Francisco, CA, USA, April 2016.

[20] Z. Cai, Z. Duan, and W. Li, "Exploiting multi-dimensional task diversity in distributed auctions for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, p. 1, 2020.

[21] Z. Duan, W. Li, and Z. Cai, "Distributed auctions for task assignment and scheduling in mobile crowdsensing systems," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 635–644, Atlanta, GA, USA, June 2017.

[22] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *UbiComp '14: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 703–714, Seattle, WA, USA, September 2014.

[23] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, "ActiveCrowd: a framework for optimized multitask allocation in mobile crowdsensing systems," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 3, pp. 392–403, 2017.

[24] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2016.

[25] H. Xiong, D. Zhang, G. Chen, L. Wang, V. Gauthier, and L. E. Barnes, "iCrowd: near-optimal task allocation for piggyback crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 2010–2022, 2016.

[26] J. Wang, F. Wang, Y. Wang, D. Zhang, L. Wang, and Z. Qiu, "Social-network-assisted worker recruitment in mobile crowd sensing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 7, pp. 1661–1673, 2019.

[27] Y. Wang, Z. Cai, G. Yin, Y. Gao, X. Tong, and G. Wu, "An incentive mechanism with privacy protection in mobile crowdsourcing systems," *Computer Networks*, vol. 102, pp. 157–171, 2016.

[28] T. Luo, S. S. Kanhere, J. Huang, S. K. Das, and F. Wu, "Sustainable incentives for mobile crowdsensing: auctions, lotteries, and trust and reputation systems," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 68–74, 2017.

[29] J. Li, Z. Cai, J. Wang, M. Han, and Y. Li, "Truthful incentive mechanisms for geographical position conflicting mobile crowdsensing systems," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 2, pp. 324–334, 2018.

[30] Z. Wang, C. Wang, X. Ye, J. Pei, and B. Li, "Propagation history ranking in social networks: a causality-based approach," *Tsinghua Science and Technology*, vol. 25, no. 2, pp. 161–179, 2020.

[31] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2020.

[32] G. Yang, Y. Li, Y. Song et al., "A real-time recommendation algorithm for task allocation in mobile crowd sensing," in *Wireless Algorithms, Systems, and Applications - 15th International Conference, WASA 2020, Proceedings, Part I*, vol. 12384 of *Lecture Notes in Computer Science*, pp. 640–652, Qingdao, China, September 2020Springer.

[33] S. Zhu, Z. Cai, H. Hu, Y. Li, and W. Li, "zkCrowd: a hybrid blockchain-based crowdsourcing platform," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4196–4205, 2020.

[34] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, and A. M'hamed, "Sparse mobile crowdsensing: challenges and opportunities," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 161–167, 2016.

[35] S. Arora, "A survey on graph neural networks for knowledge graph completion," 2020 https://arxiv.org/abs/2007.12374.

[36] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, https://arxiv.org/abs/1710.10903.

[37] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: a graph multi-attention network for traffic prediction," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence 30 Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, pp. 1234–1241, New York, NY, USA, February 2020, AAAI Press.

[38] S. Wu, W. Zhang, F. Sun, and B. Cui, "Graph neural networks in recommender systems: a survey," 2020, https://arxiv.org/abs/2011.02260.

[39] Z. Zhang, F. Zhuang, H. Zhu, Z. Shi, H. Xiong, and Q. He, "Relational graph neural network with hierarchical attention for knowledge graph completion," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020,*, pp. 9612–9619, New York, NY, USA, February 2020, AAAI Press.

[40] R. S. Sutton and A. G. Barto, *Reinforcement Learning - An Introduction. Adaptive Computation and Machine Learning*, MIT Press, 1998.

[41] Y. Dai, D. Xu, Y. Lu, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for edge caching and content delivery in internet of vehicles," in *2019 IEEE/CIC International Conference on Communications in China, ICCC 2019*, pp. 134–139, Changchun, China, August 2019, IEEE.

[42] D. Silver, J. Schrittwieser, K. Simonyan et al., "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[43] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!," in *7th International Conference on Learning Representations, ICLR 2019*, New Orleans, LA, USA, May 2019.

[44] C. Chekuri, N. Korula, and M. Pál, "Improved algorithms for orienteering and related problems," *ACM Transactions on Algorithms*, vol. 8, no. 3, pp. 1–27, 2012.

[45] K. Yang, J. Zhu, and X. Guo, "POI neural-rec model via graph embedding representation," *Tsinghua Science and Technology*, vol. 26, no. 2, pp. 208–218, 2021.

[46] Z. Ye, H. Zhao, K. Zhang, Z. Wang, and Y. Zhu, "Network representation based on the joint learning of three feature views," *Big Data Mining and Analytics*, vol. 2, no. 4, pp. 248–260, 2019.

[47] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *5th International Conference on Learning Representations, ICLR 2017, Workshop Track Proceedings*, Toulon, France, April 2017.

[48] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[49] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *KDD '11: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 316–324, San Diego, CA, USA, August 2011.