







Research Article

Dynamic Task Assignment Framework for Mobile Crowdsensing with Deep Reinforcement Learning

Yanming Fu ¹, Kangheng Qi ¹, Yuanquan Shi ², Yuming Shen ¹, Liqiang Xu ¹,
and Xian Zhang ¹

¹School of Computer Electronics and Information, Guangxi University, Nanning 530004, China

²School of Computer and Artificial Intelligence, Huaihua University, Huaihua 418099, China

Correspondence should be addressed to Kangheng Qi; qikangheng@qq.com

Received 19 December 2022; Revised 8 May 2023; Accepted 12 May 2023; Published 29 May 2023

Academic Editor: Xianfu Chen

Copyright © 2023 Yanming Fu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Task assignment is a key issue in mobile crowdsensing (MCS). Previous task assignment methods were mainly static offline assignment. However, the MCS platform needs to process dynamically changing workers and tasks online in the actual assignment process. Hence, a reliable dynamic assignment strategy is crucial to improving the platform's efficiency. This paper proposes an MCS dynamic task assignment framework to solve the task maximization assignment problem with spatiotemporal properties. First, a single worker is modeled for the Markov decision process, and a deep reinforcement learning algorithm (DDQN) is used to perform offline learning on historical task data. Then, in the dynamic assignment process, we consider the impact of current decisions on future decisions. Use the maximum flow model to maximize the number of tasks completed in each period while maximizing the expected Q value of all workers to achieve the optimal global assignment. Experiments show that the strategy proposed in this paper has good performance compared with the baseline strategy under different conditions.

1. Introduction

With the development of technologies such as 5G and microsensors, MCS has been widely used in different fields, such as intelligent transportation systems [1], environmental monitoring [2], and public safety [3]. A typical MCS system usually consists of a mobile sensing platform, workers, and task issuers. The task issuer posts tasks on the platform, and the platform distributes the collected perception tasks to workers who complete the perception tasks and get paid. Task assignment is a key issue of MCS. The current research can be divided into two categories according to the level of worker participation: the opportunistic task assignment and the participatory task assignment. In opportunistic task assignments, workers do not need to change their original trajectories, and the platform selects workers offline based on predictions of worker mobility. For example, literature [4–7] proposes different strategies for selecting a predefined number of workers to maximize the perceived quality of the

task. In participatory task assignments, workers need to generate their moving routes according to the tasks assigned by the platform, which requires candidates to report real-time locations continuously, and the system selects workers online. If a worker is selected and assigned to a specific task location, it will change the original route, move to the specified location, and receive the corresponding reward. Different kinds of literature [8–11] use different reward models for workers.

In the actual scene, the task assignment of the platform has the limitation of time and space, showing characteristics such as dynamic, strong randomness, and multistage. Most of the original MCS task assignment research has the following problems:

- (1) In MCS, workers need to collect task data at a precise location and time, so it is necessary to consider the impact of worker and task spatiotemporal information on platform task assignment

- (2) In the actual assignment process, tasks and workers change dynamically, and the platform only knows local spatiotemporal information. Therefore, the platform needs to mine the historical spatiotemporal information to optimize the task assignment strategy in decision-making
- (3) Due to the spatiotemporal continuity, the recent decision of the platform not only affects the current assignment result but also affects the following assignment result. Most of the original assignment strategies are limited to the optimal assignment in a certain stage, while the optimization goal of the platform is the optimal global assignment for the entire period

In order to solve the above problems, this paper designs a dynamic participatory task assignment framework. The framework considers the Markov decision process modeling of a single worker and introduces a deep reinforcement learning model to help the platform's decision-making at each period. In building a deep reinforcement learning model, each step of the platform's decision for a single worker is to estimate the state-action value function within the current worker's perception range and guide the worker to a place where future tasks are more intensive in its perception range. We use the DDQN [12] model to train historical data effectively, and the generated Q network is used to assist the decision-making of the platform's online assignment. At the same time, this paper designs a dynamic MCS task assignment framework, which fully considers the spatiotemporal and randomness of tasks and workers. There is a series of executable tasks within the perception range of workers in each period, and the platform maximizes the total number of tasks completed by the workers in all periods through a reasonable task assignment. Specifically, based on the maximum flow model [13] and combined with the auxiliary decision-making of the Q network, this paper realizes the dynamic online assignment of multiple periods and reaches the global maximum number of completed tasks on the platform. In summary, this paper's dynamic task assignment framework has the following characteristics:

- (1) Before task assignment begins, a Markov decision process is modeled for a single worker. We first use the DDQN model for offline training of the historical task set in crowdsensing task assignment to generate a Q network for real-time prediction
- (2) When performing online dynamic task assignments, the predicted value of the Q network and the impact of current decisions on future decisions are considered to achieve the global optimal task assignment under the platform's completion of the largest tasks in each period
- (3) According to the characteristics of MCS dynamic task assignment, this paper thoroughly mines the spatiotemporal information of historical data, estab-

lishes an MCS dynamic task assignment framework based on deep reinforcement learning, and verifies the good performance of this framework through experiments

2. Related Work

The task assignment goal is to assign perceptual tasks to eligible users, and in much literature, a lot of research has been done on task assignment algorithms for traditional MCS [14, 15]. Liu et al. [16] considered the comprehensive sensing quality of MCS to optimize the utility of the whole system. Xiao et al. [17] proposed a task assignment scheme of independent perception and cooperative perception, and the optimization goal is to minimize the average completion time of tasks. Yang et al. [18] maximize the amount of budget information for task performers under budget constraints. Liu et al. [19] proposed two dual-objective multitask assignment models, namely, FPMT and MPFT, and presented corresponding solving algorithms. Zhang et al. [20] proposed a hybrid perceptual task model with the goal of maximizing task completion and perceptual coverage. Zhang et al. [21] used a greedy heuristic algorithm and a genetic algorithm to solve the problem based on two optimization objectives in the vehicle crowdsensing system. Li et al. [22] established an optimization model for crowdsourcing task assignment in heterogeneous spaces: maximizing task coverage and minimizing incentive cost and designed a greedy swarm intelligence optimization algorithm to solve two optimization objectives.

In studies where workers select a single MCS task, task assignment has specific goals and constraints. For example, the literature proposes different recruitment strategies to select a predefined number of workers to maximize the perceived quality of the task [4–6]. Zhang et al. [23] chooses a minimum number of workers to ensure a certain degree of perceived quality. Ji et al. [24] propose a mobile crowdsensing system with social awareness and design an improved MOEA/D algorithm to achieve the Pareto optimal solution set. As the number of MCS tasks increases and the tasks are interrelated, some studies consider the overall utility of multiple concurrent perception tasks. For example, Song et al. [25] and Wang et al. [26] both proposed multitask assignment algorithms that maximize the system's overall utility when tasks share a limited incentive budget. The multitask assignment strategy proposed by Wang et al. [27] is aimed at optimizing the overall utility when multiple tasks share a constrained worker pool.

Some studies collect location-related data centrally without any time constraints. Shah-Mansouri and Wong [28] used the auction mechanism to maximize the profit of a single-task platform without a time limit while providing satisfactory rewards for workers. Zhou et al. [29] used a single-objective optimization method with budget constraints to maximize task quality efficiency under the premise of considering employee reputation and no time constraints. However, in practice, users need to complete these tasks before certain deadlines, so we need to consider the impact of time on task assignments. Some recent studies

have focused on centralized task assignment schemes with time constraints. For example, Cheung et al. [30] considered collecting time-sensitive and location-dependent perception data by multiple users. They proposed a distributed algorithm to help users determine their task choices and mobility plans. Estrada et al. [31] studied the trade-off between quality, budget, and time constraints of perception tasks over a period of time. They provided a service computing framework for task assignments with time and location constraints.

In recent years, reinforcement learning has been used in the MCS to make a range of decisions in uncertain environments. Ji et al. [32] construct a dynamic task allocation model and proposes a Q-learning-based hyperheuristic evolutionary algorithm to maximize the average perceived quality of all tasks in each period. Akter et al. [33] proposed a deep Q-learning-based algorithm to determine the assignment of tasks and workers and iteratively used the asymmetric travelling salesman (ATSP) heuristic to find the task completion order of workers. Wang et al. [34] proposed a privacy-enhanced multiregional task assignment strategy (PMTA) for Healthcare 4.0 using deep differential privacy, deep Q network, federated learning, and blockchain to effectively protect the privacy of tasks and patients and obtain better system performance. Tao and Song [35] tried to use deep reinforcement learning methods to find a more efficient task assignment solution and used DDQN to solve the task assignment problem with time windows. Wang et al. [36] proposed a blockchain-based secure data aggregation strategy (BSDA) for edge computing-enhanced IoT. BSDA adopts three important mechanisms to prevent privacy leakage and develops a deep reinforcement learning method for energy-efficient data aggregation. Han et al. [37] proposed a real-world-oriented multitask assignment method based on multiagent reinforcement learning. This method fully considers worker and task heterogeneity. It is based on an improved soft Q-learning method that enables workers as agents to learn multiple solutions independently, which optimizes the perceptual quality of tasks. The above literature mainly applies reinforcement learning to task assignment optimization with time windows. The platform finds the optimal assignment strategy offline after knowing all the information about the workers and tasks. However, in actual task assignments, workers and tasks are dynamically changed. This paper first uses reinforcement learning in crowdsensing task assignments to train offline on historical data and provide decision-making for dynamic platform assignments.

3. Model Description and Problem Definition

Assuming that there is a batch of worker sets and task sets with spatiotemporal attributes in the MCS platform, the platform needs to assign appropriate tasks to each worker dynamically. In order to simplify the model, this paper divides the entire task assignment stage of the platform into equal small periods represented by $T = \{t_1, t_2, \dots, t_k, \dots, t_h\}$. Workers in all periods are represented by the set $W = \{w_1, w_2, \dots, w_i, \dots, w_m\}$, and each worker has the following attributes:

$w_i = \langle \text{lat}_i, \text{lng}_i, t_i, t_{ik} \rangle$. Among them, lat_i and lng_i represent the current latitude and longitude coordinates of the worker and t_i represents the period when the worker joined the platform. Workers will continue to participate in task assignment after joining until reaching the deadline t_h , so t_{ik} represents the period the worker is in, where $t_i \leq t_{ik} \leq t_h$. The task in all periods consists of the set $L = \{l_1, l_2, \dots, l_j, \dots, l_n\}$, and the task has the following attributes: $l_j = \langle \text{lat}_j, \text{lng}_j, t_j, n_j \rangle$. Among them, lat_j and lng_j represent the latitude and longitude of the task and t_j represents the period when the task is posted to the platform, where $t_1 \leq t_j \leq t_h$. In this article, the task is time-sensitive, it needs to be completed within the period t_j , and n_j represents that a task is done by at most n_j different workers.

Different from traditional task assignments, tasks and workers are dynamic in the task assignment process of this paper, and the platform only has information on workers and tasks in the current period. In the period t_k , the set of workers is composed of $t_{ik} = t_k$ workers, using the set $W^{t_k} = \{w_1^{t_k}, w_2^{t_k}, \dots, w_i^{t_k}, \dots, w_m^{t_k}\}$, where $m' \leq m$; the task set consists of $t_j = t_k$ tasks, using the set $T^{t_k} = \{t_1^{t_k}, t_2^{t_k}, \dots, t_i^{t_k}, \dots, t_n^{t_k}\}$, where $n' < n$. Each worker can only be assigned a maximum of one task in each period, if the worker $w_i^{t_k}$ is currently assigned a task, then $w_i^{t_k} = 1$; otherwise, $w_i^{t_k} = 0$. Figure 1 shows the system model of dynamic task assignment in each period. Firstly, the platform obtains information about workers and tasks in the current period. Then, the platform will assign each worker a task within the maximum perception radius D_{\max} based on the current spatiotemporal information of workers and tasks. The assignment result will be returned to the worker. After receiving the task instruction, the worker moves to the task location to complete the task in this period and uploads the perception information to the platform. Finally, the platform will return the perception result to the task issuer, update the spatiotemporal information of the worker and the task, and enter the allocation of the next period. In particular, to simplify the model, this paper considers the ideal case where the tasks appearing in each period of the MCS are time-sensitive and the value of the tasks is much larger than the cost of moving the workers within their perceived range. Therefore, we do not consider worker consumption and assume that all tasks within the worker's perceived range are equal, while the main goal of the platform is to guide the worker to complete more tasks. Suppose there is no task that can be assigned to the worker within the perception range of the current period. The worker will stay at the original location or move a short distance randomly in the perception range, waiting for the assignment of the next period.

Therefore, we define the cumulative number of tasks completed by the platform as the sum of the number of tasks completed in multiple periods throughout the assignment:

$$\sum_{k=1}^h \sum_{i=1}^{|W^{t_k}|} w_i^{t_k}. \quad (1)$$

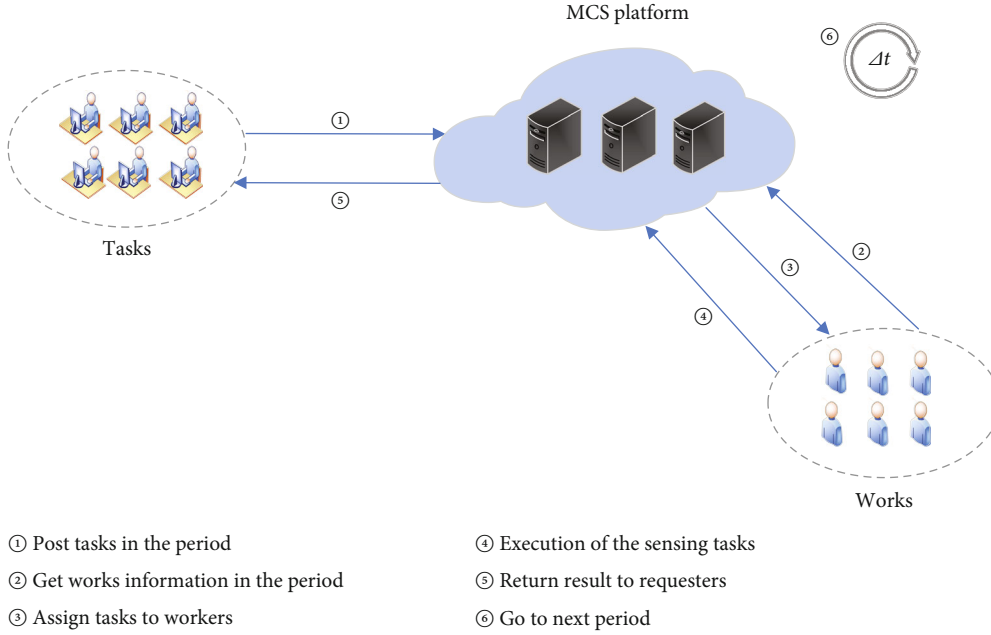


FIGURE 1: Dynamic task assignment model of MCS platform in each period.

In Equation (1), $|W^{t_k}|$ represents the total number of workers in the worker set W^{t_k} in the period t_k , and $w_i^{t_k}$ represents the tasks completed by the workers $w_i^{t_k}$ in the period t_k . In the actual assignment process, the platform needs to consider the positions of workers and tasks in real time. In order to ensure that workers can complete perception tasks within a period, this paper defines the maximum task perception radius of workers:

$$d \leq D_{\max}. \quad (2)$$

The distance d between the worker and the task in Equation (2) is obtained from the latitude and longitude coordinates of the worker and the task according to the haversine formula:

$$d = 2r \arcsin \left(\sqrt{\sin^2 \left(\frac{\text{lat}_i - \text{lat}_j}{2} \right) + \cos(\text{lat}_i) \cos(\text{lat}_j) \sin^2 \left(\frac{\text{lng}_i - \text{lng}_j}{2} \right)} \right), \quad (3)$$

where r is the radius of the earth and lat_i , lng_i , lat_j , and lng_j are the latitude and longitude coordinates of workers and tasks, respectively.

The goal of MCS is to maximize the total number of completed tasks, from which we obtain the optimization objective and spatiotemporal constraints for dynamic task assignment:

$$\max \sum_{k=1}^h \sum_{i=1}^{|W^{t_k}|} w_i^{t_k}. \quad (4)$$

Subject to:

$$|LW_j| \leq n_j, \quad (5)$$

$$t_{ik} = t_k = t_j, \quad (6)$$

$$d_i \leq D_{\max}. \quad (7)$$

Equation (4) indicates that the optimization goal of the platform is to maximize the number of completed tasks in multiple periods. In Equation (5), $|LW_j|$ represents the number of workers who complete task l_j , which is less than n_j of the task. Equation (6) is the time limit, where $t_{ik} = t_k = t_j$ means that the worker $w_i^{t_k}$ needs to arrive at the task location within the t_k period to complete the task l_j . Equation (7) is the space limit, which means that the task perception radius d_i of worker w_i in each period is smaller than D_{\max} .

4. Problem Solving

The most important feature of dynamic task assignment is that the platform does not have complete spatiotemporal information about workers and tasks throughout the period so that the current decision-making will impact future decision-making. Therefore, we consider the dynamic task assignment problem as a multistage sequential decision-making problem. Firstly, the Markov decision process is modeled for a single worker and then used the improved DDQN algorithm to train historical data to generate a Q network. Finally, the overall framework of MCS dynamic task assignment is designed to achieve reasonable task matching in all periods.

4.1. Markov Decision Process Modeling from a Single Worker Perspective. A Markov decision process (MDP) is a discrete-time stochastic control process that provides a mathematical model for decision-making problems. Most of the literature conducts MDP modeling on the entire MCS platform [31, 33]; however, this will make the model

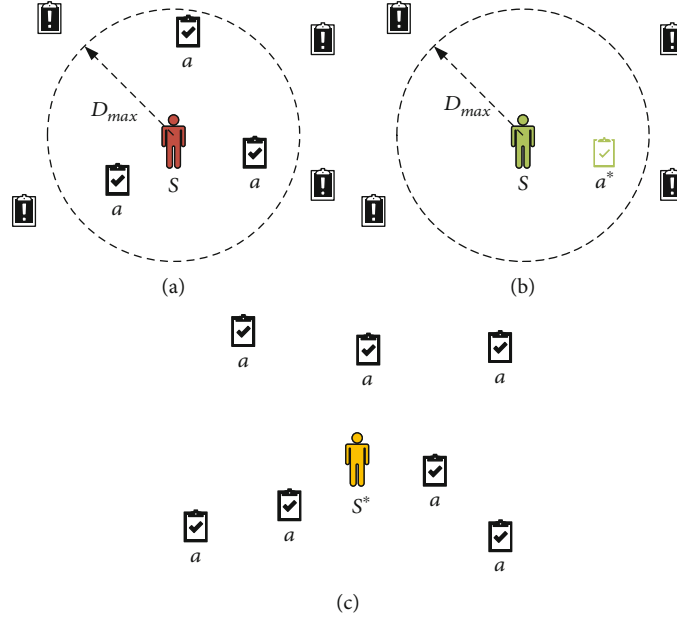


FIGURE 2: Action search: (a) in state s , (b) in virtual action a^* , and (c) in the initial state S_0^* of training the Q network.

complex, and it is difficult to fully learn the information in the environment. In this paper, MDP modeling of task assignment is carried out from the perspective of a single worker, and a single worker is regarded as an agent, which significantly simplifies the definitions of state transitions, actions, and rewards. The relevant definitions are as follows:

State s is defined as the information of a single worker in the system, that is, $s = \langle \text{lat}_i, \text{lng}_i, t_{ik} \rangle$, where lat_i and lng_i represent the worker's latitude and longitude coordinates, and t_{ik} represents the current period of the worker. When $t_{ik} > t_h$, s is a termination state. Specially, when training the Q network, in order to better explore the historical environment and reduce the influence of the randomness of the initial position, the agent is not limited by the perception range in the initial state. The initial state is represented by $S_0^* = \langle \text{lat}_{i0}^*, \text{lng}_{i0}^*, t_{ik0}^* \rangle$, where $\langle \text{lat}_{i0}^*, \text{lng}_{i0}^* \rangle$ is a virtual location label, and the agent can select all the tasks in the initial period t_{ik0}^* , so that it can fully explore the historical task expected value.

Action a is defined as the task information assigned to the worker by the platform, and the worker can only complete the task in the period, so we set the action as the position of the task, that is, $a = \langle \text{lat}_j, \text{lng}_j \rangle$, where lat_j and lng_j indicate the latitude and longitude coordinates of the task. The feasible action search of the agent in the state is shown in Figure 2: (a) In state s , all tasks within the range of the agent perception are feasible actions. (b) If there is no executable task within the perception range of the agent, a virtual task (virtual action a^*) with return of 0 is constructed, and the worker considers staying in place or going to any point within the perception range, waiting for the allocation of the next period. (c) Specially, in the initial state S_0^* of training the Q network, all tasks in the current period are the feasible actions of the agent.

The worker goes to the task location to perform the task and transitions to the next state $s' = \langle \text{lat}_i', \text{lng}_i', t_{ik}' \rangle$, where lat_i' and lng_i' are the latitude and longitude of the task, $t_{ik}' = t_{ik} + 1$. Reward $R(s, a)$ represents the worker's benefit completing action a in state s . In this paper, we assume that each task is equivalent and want workers to travel to task-dense regions to avoid no feasible tasks within the perception range of the following period. Therefore, the reward $R(s, a)$ is set to a constant R , whose magnitude is determined by the number of tasks within the worker's perception range. Specially, the reward is 0 when the action a^* is taken. In the initial state S_0^* , since the agent is not limited by the range of perception, the rewards of all actions of the agent are equal.

The state-action function $Q(s, a)$ represents the expectation of the total benefit that the worker can obtain in the future when he takes action a in state s :

$$Q(s, a) = E \left[\sum_{t=0}^T \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a \right], \quad (8)$$

where T is the number of steps for state s to reach the terminal state, and γ is the discount coefficient of future rewards.

Policy π refers to a probability distribution based on a set of behaviours in a certain state. In this paper, model-independent reinforcement learning is used to learn the optimal policy through the interaction between the agent and the environment to maximize the expected cumulative return. The greedy policy with respect to a learned $Q(s, a)$ is given by

$$\pi(s) = \arg \max_a Q(s, a). \quad (9)$$

4.2. Offline Q Network Training Based on Improved DDQN. Based on the definition of MDP, this paper adopts the reinforcement learning algorithm to train the historical

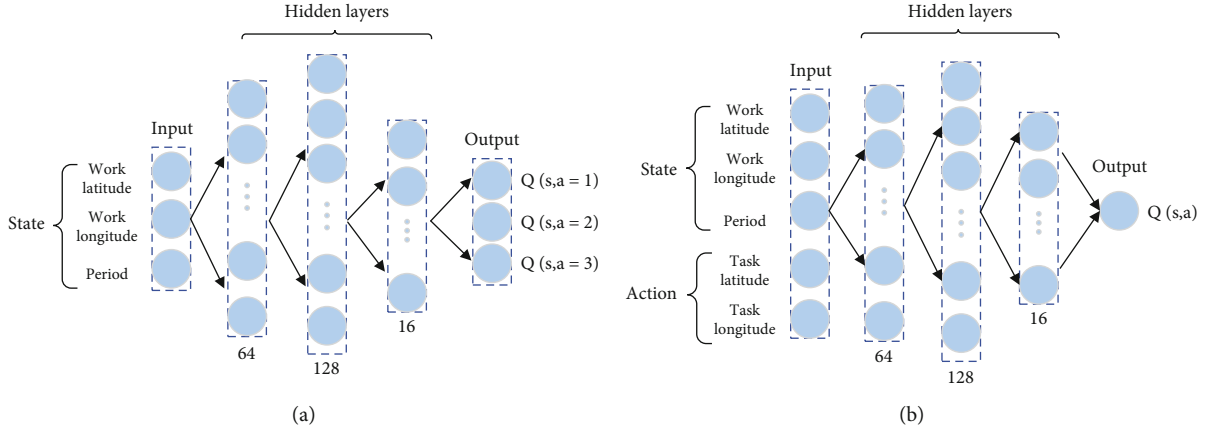


FIGURE 3: Deep neural network structure: (a) DDQN and (b) this paper.

data. Q-learning [38] is a widely used reinforcement learning method that mainly focuses on estimating the value function for each state-action pair with a Q value table. For any state $s \in S$ and action $a \in A$, Q-learning predicts the value of the state-action pair by iteratively updating

$$Q(s, a)^{\text{New}} \leftarrow Q(s, a) + \alpha \left(R + \gamma \max_{a'} Q(s', a') - Q(s, a) \right), \quad (10)$$

where α is the learning rate, γ is the discount factor, and R is the reward for the state transition from s to s' after taking action a . $\max_{a'} Q(s', a')$ is the largest Q value function among all possible actions in the new state s' .

Since the dynamic task assignment needs to use space-time-based continuous state space S and action space A , the dimension of the state space is large. This paper adopts a single worker Markov decision process solution based on DDQN [12] model and approximates the value function in Q-learning with a deep neural network to find the optimal policy. On the neural network architecture, traditional DQN assumes a small discrete action space, using states as inputs and multiple outputs corresponding to the action values of fixed actions, as shown in Figure 3(a). Due to the huge action space in the training historical data and the action space that will change continuously over time, $Q(s, a)$ cannot be enumerated. Therefore, we improved the network structure of DDQN in combination with literature [39], and Figure 3(b) shows the structure of a deep neural network. The input consists of state s and action a . State s contains the latitude and longitude coordinates of the worker and the period. Action a contains the latitude and longitude coordinates of the task. Combined with literature [40], the hidden layer uses a three-layer fully connected network, and the number of neurons is set to 64, 128 and 16, respectively. The output is the state-action value $Q(s, a)$.

This paper uses the interaction of a single agent with the historical data environment to explore the expected value of tasks in different locations. The agent starts from the virtual

initial state S_0^* and generates the sample sequence $\langle s, a, r, s', \text{info} \rangle$ required for training the neural network by interacting with the environment of the historical task set and stores it in the replay memory. When the replay memory is full, randomly select a minibatch of data for training. In the Q network training framework based on the DDQN algorithm, the minibatch update via backpropagation is essentially a step in solving a regression problem with the following loss function:

$$\text{Loss}(\theta) = (y - Q(s, a, \theta))^2, \quad (11)$$

where y is calculated from

$$y = \begin{cases} r, & s' \text{ is the termination state,} \\ r + \gamma Q^* \left(s', \arg \max_{a' \in A'} Q(s', a', \theta); \theta^* \right), & s' \text{ is not a termination state.} \end{cases} \quad (12)$$

Finally, after the training is complete, the information of the worker's state-action pair is used as input, and the predicted Q value of the state action is output. Combined with the characteristics of MCS dynamic task assignment, we designed a Q network training algorithm based on the DDQN model. The specific steps are as follows:

Algorithm 1 first selects the historical dataset H as the historical environment of the task and has all the spatio-temporal information of the environment during training. The worker starts from the initial state, explores the historical environment of the task, generates a training sequence $\langle s, a, r, s', \text{info} \rangle$, and puts it into the replay memory D . After the replay memory D is full, the neural network trained and updated the replay memory. When the worker reaches the deadline, it becomes terminated and enters the next loop. Through continuous training, the algorithm converges and outputs the evaluation network Q .

4.3. A Dynamic Task Matching Strategy for Multiple Workers in Each Period. In the actual task assignment process, there may be multiple workers and tasks in each period. The optimization goal of the MCS platform is to maximize the

```

Input: Historical data set H, replay memory D, maximum training episodes N, a constant Z, initialization evaluation network Q and target network Q*
Output: Evaluate network Q
1: For step from 1 to N do
2:   Initialized worker state  $S_0^* = \langle lat_{i_0}^*, lng_{i_0}^*, t_{ik0}^* \rangle$ 
3:   while  $s'$  is not the termination state do
4:     if  $s$  is the initial state  $S_0^*$  then
5:       Take the task with period equal to  $t_{ik0}^*$  in the historical data set H as the
       action set of the current state  $s$ 
6:     else
7:       Obtain the action set of the current state  $s$  in the historical data set H
       according to the spatiotemporal constraints of Equation (6) and Equation (7)
8:     end if
9:     if the action set of  $s$  is empty then
10:      The worker executes the virtual task  $a^*$ , the state transitions to  $s'$ , and
      the reward  $r$  is 0
11:      Store  $\langle s, a, r, s', info \rangle$  in the cache memory, where  $s'$  is the next state,
       $r$  is the reward, and  $info$  is whether  $s'$  is the termination state
12:    else
13:      Take  $\phi(s, a)$  as input to get the  $Q(s, a, \theta)$  value for each state-action
      pair
14:      Use the  $\epsilon$ -greedy method to select the corresponding action  $a$  in the
      output of the current  $Q(s, a, \theta)$  value
15:      Get  $s', r, info$  according to action  $a$ . and store  $\langle s, a, r, s', info \rangle$  in the
      replay memory
16:    end if
17:    if replay memory D is full then
18:      Cover a piece of data in D and extract a mini-batch to randomly sample
       $\langle s, a, r, s', info \rangle$  for learning
19:      Calculate the target value  $y$  according to Equation (12)
20:      Gradient descent update of evaluation network Q parameters according
      to the loss function of Equation (11)
21:      Update the target network  $Q^*$  parameters every Z step  $\theta^* = \theta$ 
22:    end if
23:     $s = s'$ 
24:  end while
25: end for

```

ALGORITHM 1: Q network training algorithm based on DDQN model.

number of completed tasks, which can be based on the maximum flow model [13] to maximize the number of completed tasks in each period. However, due to the continuity of time and space, the platform's current task assignment decision will affect the following assignment result, and the traditional maximum flow algorithm may fall into the optimal local period. Therefore, we combine the prediction value of the Q network to construct the worker task matching of each period into a maximum Q value maximum flow model to achieve the optimal matching of the entire period. Figure 4 shows the model diagram of the network. The first layer and the tail layer are the source node and the tail node, respectively, and the middle layers contain the platform's worker nodes and task nodes in the current period. Each edge has two attributes: capacity C and cost B . For the edge between the source node and the worker node, the capacity of each edge is 1, reflecting that each worker is assigned at most 1 task per period; the cost of the edge is 0, because the edge between the source node and the worker node is

only used for inflow. For the edge between the worker node and the task node, the platform assigns the corresponding edge in the graph by searching for the task set within the maximum perception range under the current spatiotemporal information of each worker. The capacity of each edge is 1, and the cost of the edge is $Q(s, a)$. For edges between task nodes and tail nodes, each edge has a capacity of n_j , which reflects the maximum number of a task can be completed by different workers; the cost of the edge is 0, because edges between task and tail nodes are only used for outflow.

This paper proposes a maximum Q value maximum flow matching strategy (MaxflowQ) in each period, as shown in Algorithm 2. The platform maximizes the sum of $Q(s, a)$ values of workers while matching worker tasks in each period to the maximum flow, so as to achieve the overall optimization of dynamic assignment. Algorithm 2 consists of two parts: one is to construct the flow network of the model, and the other is to find the optimal

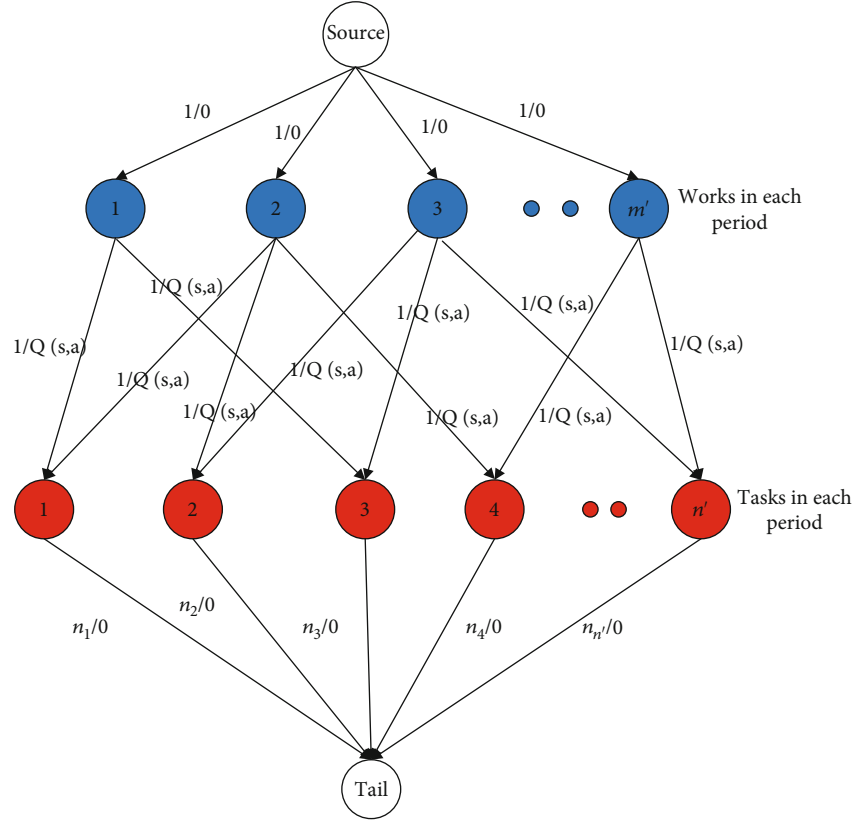


FIGURE 4: Maximum Q value maximum flow model.

Input: The platform worker set W^{t_k} in the current period, the task set $A_i^{t_k}$ within the perception range of each worker, the $Q(s, a)_i^{t_k}$ set of workers $w_i^{t_k}$.

Output: Worker task matching set WL^{t_k} in the current period.

- 1: According to W^{t_k} , $A_i^{t_k}$, $Q(s, a)_i^{t_k}$ constructs the flow graph $G = (V, E, C, B)$
- 2: Initialize flow f to 0
- 3: **while** there exists an augmenting path in the residual network G_f **do**
- 4: Select an augmenting path p^* with the largest Q value
- 5: $c_f(p^*) = 1$
- 6: Augment flow f along p^* with $c_f(p^*)$
- 7: Update residual network G_f
- 8: Save worker task matches
- 9: Update matching set WL^{t_k}
- 10: **end while**

ALGORITHM 2: Maximum Q value maximum flow matching strategy (MaxflowQ).

solution in the flow network. The platform constructs a flow network $G = (V, E, C, B)$ based on the worker set W^{t_k} in the current period, the task set $A_i^{t_k}$ within the perception range of each worker, and the $Q(s, a)$ set of workers $w_i^{t_k}$, where C is the capacity of each edge and B is the cost of each edge, as shown in Figure 4. Subsequently, the optimal solution is found in the flow network. The procedure is as follows: firstly, initialize the flow graph G and then greedily select the augmented path p^* from node source to node tail in the residual network

G_f with maximum cost, along p^* with the capacity of $c_f(p^*)$ increases flow f until there are no additional paths in the remaining network G_f . Finally, the algorithm outputs the worker task matching set WL^{t_k} for the current period.

4.4. MCS Dynamic Task Assignment Solution Framework. Through the above discussion, we construct the overall framework to solve the dynamic task assignment of the MCS platform, as shown in Figure 5.

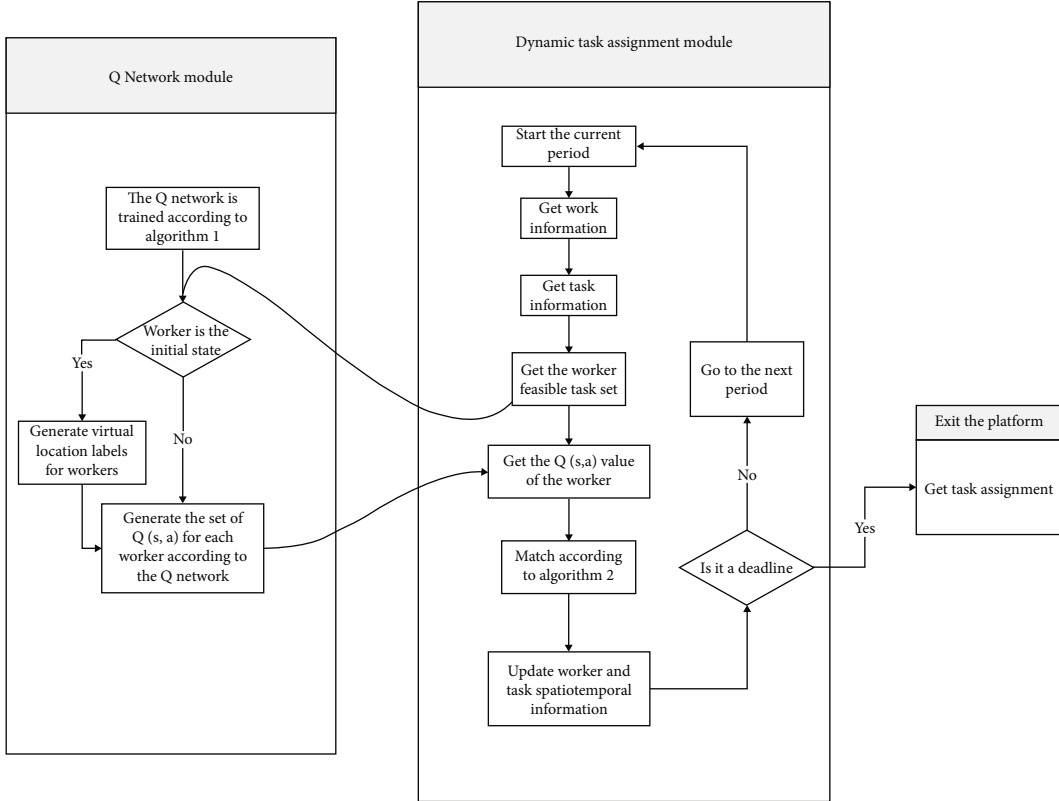


FIGURE 5: Dynamic task assignment solution framework.

The task assignment system of the MCS platform consists of two modules: the Q network module and the dynamic task assignment module. Before the task assignment starts, the MCS platform uses historical data to train offline according to Algorithm 1 to generate a Q network for decision-making assistance. During task assignment, the MCS platform uses the dynamic task assignment module to match workers and tasks in real time at each period, including the following three steps: (1) The platform first obtains the spatiotemporal information of the worker set W^{t_k} and the task set L^{t_k} in the current period and searches for the task set $A_i^{t_k}$ within the perception range of each worker $w_i^{t_k}$ based on the spatiotemporal information. (2) The platform generates the state-action pairs of the worker $w_i^{t_k}$ according to the task set $A_i^{t_k}$ of each worker $w_i^{t_k}$. The Q network module is then called to generate $Q(s, a)_i^{t_k}$ values for all actions in the current state of worker $w_i^{t_k}$, which is used to assist the platform's task assignment decisions. (3) Finally, the platform will construct a flow network $G = (V, E, C, B)$ according to Algorithm 2 and generate the optimal worker task matching for the current period t_k . After receiving the task information sent by the platform, the workers move to the corresponding location to complete the task and enter the allocation for the next period. When the deadline is reached, the platform completes the assignment and exits the dynamic task assignment framework. From this, we construct the dynamic task assignment of the MCS platform as Algorithm 3.

5. Experimental and Analysis

5.1. Experimental Dataset Selection and Processing. In order to verify the effectiveness of the framework, two real datasets, the Gowalla dataset [41] and the Foursquare dataset [42], were selected for model training and testing. The Gowalla dataset contains check-in records which contain a large number of users, including user ID, check-in time, latitude, longitude, and location ID. The Foursquare dataset contains check-ins collected in New York City and Tokyo, and each check-in contains the following records: user ID, venue ID, venue category ID, venue category name, latitude, longitude, time zone offset in minutes, and UTC time.

In the experiment, the check-in data from 9:00 to 12:00 in New York City for two months in two datasets were selected as the training set, and the data of different days after the training set was used as the test set. The latitude and longitude of the check-in records in the dataset are regarded as the location of the task set in the MCS, and a randomly different maximum number of n_j is generated for each task. At the same time, in order to simplify the model, the check-in data of the Gowalla dataset and the Foursquare dataset are divided into 24 periods and 40 periods with the same interval, and the check-in data of the same place in the same period is deleted. In addition, the initial check-in location and period of workers with different IDs are selected as the initial period and location of workers for task assignment testing. At the same time, in the two datasets, there are obviously hot and cold areas in

Input: Task set L , worker set W , evaluation network Q
Output: Complete task set WL

- 1: **for** t_k from 1 to h **do**
- 2: Obtain the worker set W^{t_k} and the task set L^{t_k} in the current period
- 3: **for** $w_i^{t_k}$ from 1 to $|W^{t_k}|$ **do**
- 4: Obtain the feasible task set $A_i^{t_k}$ of worker $w_i^{t_k}$ in L^{t_k} according to the spatiotemporal constraints of Equation (6) and Equation (7)
- 5: **if** $A_i^{t_k}$ is empty **then**
- 6: Workers stay in place or move a short distance at random in perception range, waiting for the next period to be allocated
- 7: Continue
- 8: **end if**
- 9: **if** Worker $w_i^{t_k}$ is the initial state **then**
- 10: Set the value of the Q network input of the worker $w_i^{t_k}$ as the virtual location label $\langle lat_{i0}^*, lng_{i0}^* \rangle$
- 11: **end if**
- 12: Generate state-action pairs for workers $w_i^{t_k}$
- 13: Input the state-action pair to the Q network generated by Algorithm 1, and obtain the $Q(s, a)_{i^{t_k}}$ set of workers $w_i^{t_k}$
- 14: **end for**
- 15: Generate worker task matching set WL^{t_k} according to Algorithm 2
- 16: Update worker and task information
- 17: Proceed to the next period's assignment
- 18: **end for**

ALGORITHM 3: MCS dynamic task assignment algorithm

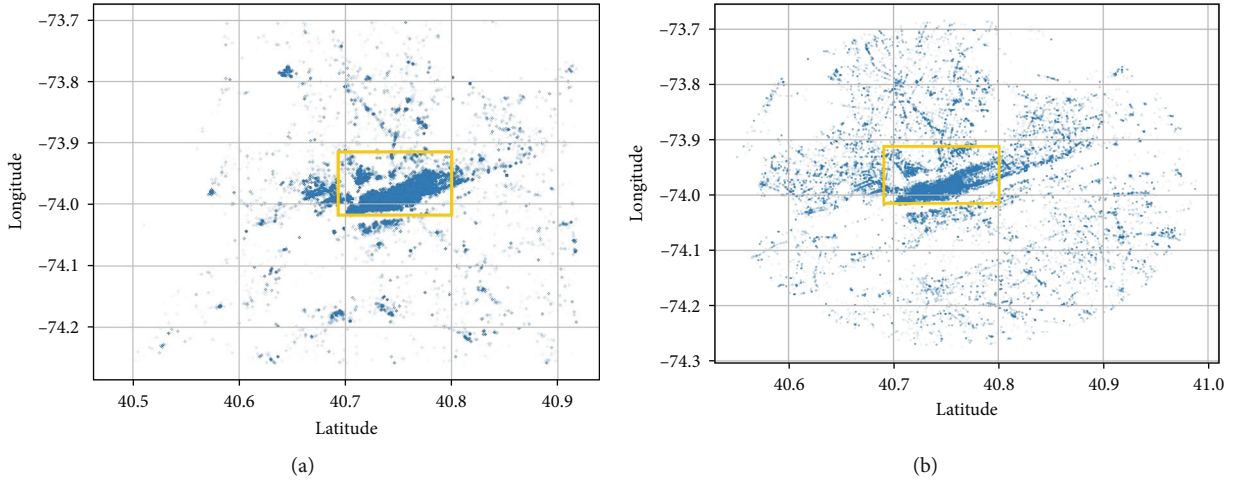


FIGURE 6: Schematic diagram of the selected area: (a) Gowalla and (b) Foursquare.

the check-in locations in New York City, so a rectangular area of about $15 \text{ km} \times 10 \text{ km}$ was intercepted, as shown in Figure 6.

5.2. Experimental Environment and Parameter Settings. The experiment is implemented based on python and uses PyTorch to build a deep neural network model. The choice of parameters when training a deep reinforcement learning network may affect the solution results. There are general principles to follow and related literature for reference [12, 39]. The randomly chosen action probability ϵ decreases from 0.2 to 0.1 during training. The discount factor measures the

weight of the subsequent state-action value to the total return, so the value is generally close to 1, and $\gamma = 0.95$ in the experiment. The learning rate $\alpha = 0.001$, the replay memory capacity $RM = 50000$, the sampling minibatch is 64, the network parameter θ, θ^* adopts a random initialization strategy, the target network parameter θ^* delay update steps $Z = 300$, the rule activation function is used, and the adaptive moment estimation algorithm is selected. At the same time, in order to eliminate the dimension, the data obtained from the input layer of the neural network is normalized. The specific parameter settings of Q network training and dynamic task assignment test are shown in Table 1.

TABLE 1: Experiment definition parameters.

Parameters	Settings
RM (capacity of the replay memory)	50000
ε (probability of random selection)	0.2-0.1
Z (steps to update the target network)	300
γ (discount factor)	0.95
α (learning rate)	0.001
Minibatch size	64
Sliding window	50
N (maximum training episodes in Algorithm 1)	50000
n_j (the maximum number of a task completed)	1-3
Periods in Gowalla	24
Periods in Foursquare	40
Number of workers in Gowalla	[60,70,80,90,100]
Number of workers in Foursquare	[30,40,50,60,70]
Task test sets days in Gowalla	[7,9,10,12]
Task test sets days in Foursquare	[10,12,15,20]
D_{\max} (perception radius)	[1 km, 2 km, 3 km]

5.3. Evaluation of Q Network

5.3.1. Parameter Selection of Training. In training on historical data, we hope to learn optimal policies through worker-environment interactions that maximize the expected cumulative return. Therefore, this paper uses the improved DDQN model to learn the optimal strategy. The DDQN model is sensitive to the selection of hyperparameter such as the learning rate. To explore the effect of different learning rates on the optimal learning of DDQN models, the experiment sets $t_{ik_0}^*$ in the worker's initial state S_0^* to 1 in each iteration and compares the worker's reward value in the same initial state. The learning rate is set 0.01, 0.001, and 0.0001, respectively. The maximum perception radius D_{\max} is set to 2km, and the maximum episode is set to 30,000.

Figure 7 shows the evolution of the reward value during training the historical data in Gowalla and Foursquare, using a sliding window every 50 episodes to calculate the reward value curve. As can be seen from the figure, among the three learning rate settings, the highest reward value for training is achieved with a learning rate of 0.001. The learning rate of 0.0001 converges the slowest. Although the reward rises fastest at a learning rate of 0.01, it has worse values and fluctuations in convergence than the case of a learning rate of 0.001. The stability of the training process and the better strategy is more important than the training speed. Therefore, we set the learning rate to 0.001 in the Q network training.

5.3.2. Q Network Train and Analysis. The experiment uses Algorithm 1 to train the historical data. The converged Q network can be used to predict the expected Q values of dynamically changing worker task pairs in real envi-

ronments. The initial state of the worker is set to $S_0^* = \langle \text{lat}_{i_0}^*, \text{lng}_{i_0}^*, t_{ik_0}^* \rangle$. Since workers mostly appear at the initial time of the platform allocation distribution in practice, at each iteration, $t_{ik_0}^*$ randomly selected integers from 1 to 3. The maximum perception radius D_{\max} for a single worker is set to 2 km, and the number of network training is set to 50,000 episodes.

Figure 8 shows the changes in the value of the $\text{Loss}(\theta) = (y - Q(s, a, \theta))^2$ function value during the training process in the Gowalla dataset and the Foursquare dataset, respectively. For the training results, a sliding window every 50 times was used to calculate the loss value curve. It can be seen from the figure that the function value has a good convergence effect. During the training process, the function value will fluctuate due to the agent's exploration of the environment and acquiring new environmental information. With the continuous increase of training times, the fluctuation range of the function value will gradually become smaller, and there is a decreasing trend.

In this paper, we expect workers to travel to task-intensive areas based on the predicted values of the Q network to avoid no feasible tasks within the perception range of the next period. To verify the predictive effectiveness of the Q network, we compare the expected values of different locations in the same period. The state value function $V(s)$ represents the expected cumulative reward that workers will be able to obtain by following policy π from state s until the end of the state. Assuming that a greedy strategy based on state-action value $Q(s, a)$ is always used in the assignment process, then the state value function $V(s)$ is shown in

$$V(s) = Q(s, \pi(s)) = \max_{a \in A} Q(s, a). \quad (13)$$

Figure 9 shows the distribution of the V value (the maximum Q value of all possible action-state pairs in the current state) for different states in a certain period in the Gowalla and the Foursquare datasets, respectively. It can be seen that the V values of different states in the same period are mainly affected by space. The part with a large red V value in the figure corresponds to the area with denser tasks in the lower right corner of Figure 6, while the blank area and the blue V value of the small part in the figure correspond to the blank area and the area with sparse data in Figure 6. It shows that the Q network has good predictability for the hot and cold distribution of tasks in different regions.

In summary, it shows the good performance of Q network in this paper. The Markov decision process is modeled for a single worker, and the Q network generated by training evaluates its own state to find a better strategy. However, when the platform performs dynamic allocation, multiple workers will affect each other. Then, the experiment will perform the joint dynamic assignment of different workers in the test task set to simulate the assignment in the actual scene and verify the effectiveness of the proposed framework.

5.4. Evaluation of Dynamic Task Assignment System Framework. In this section, we evaluate the performance of the overall dynamic task assignment system framework in

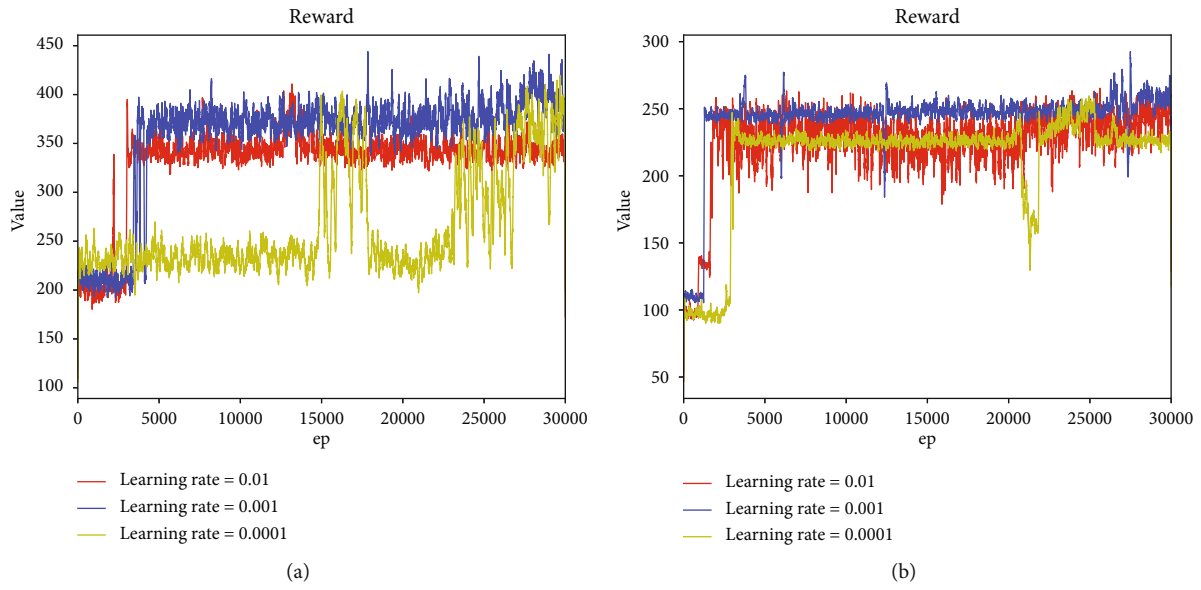


FIGURE 7: Reward value in different learning rates: (a) Gowalla and (b) Foursquare.

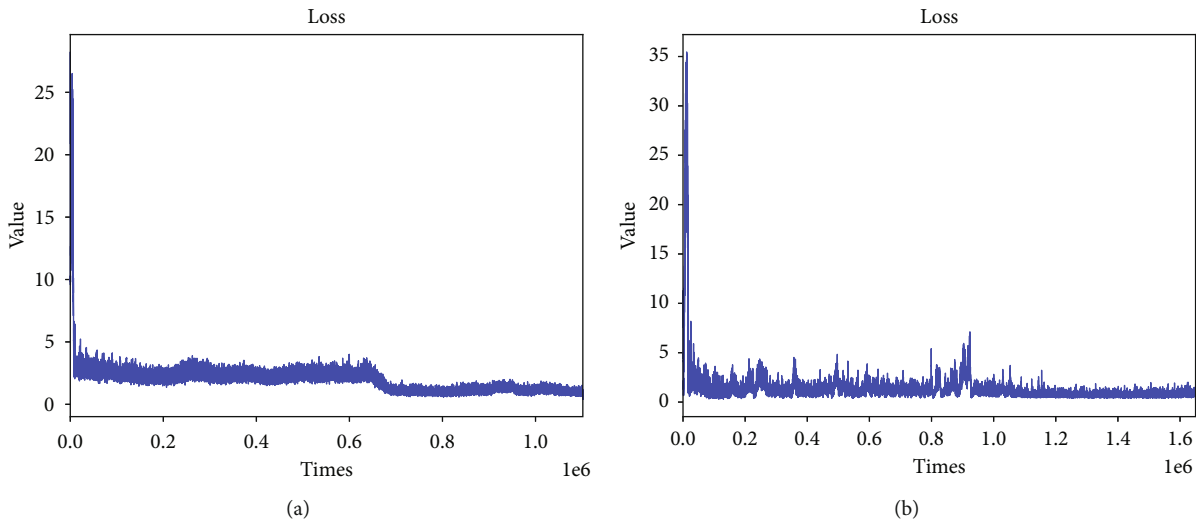


FIGURE 8: Q network loss function value: (a) Gowalla and (b) Foursquare.

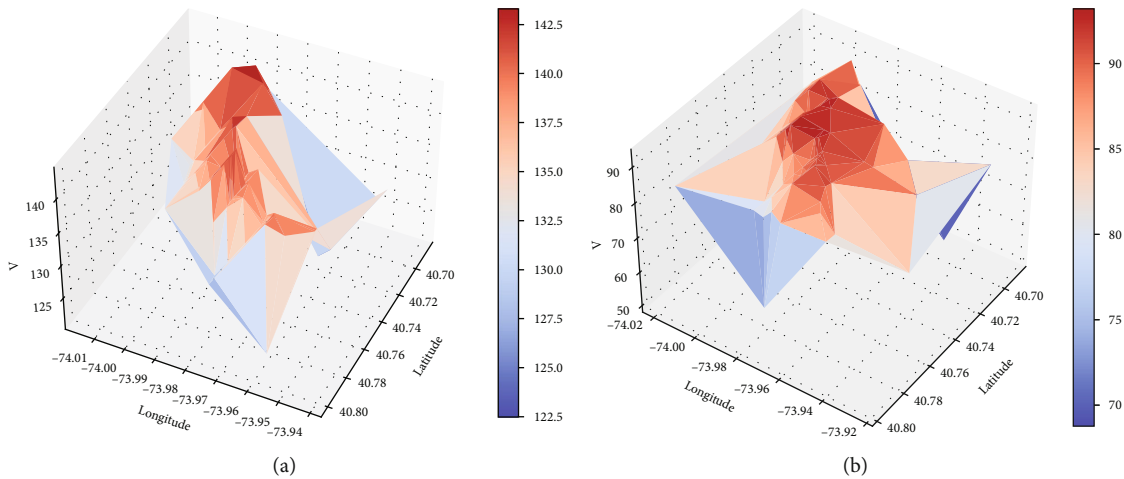


FIGURE 9: V values of different states in a certain period: (a) Gowalla and (b) Foursquare.

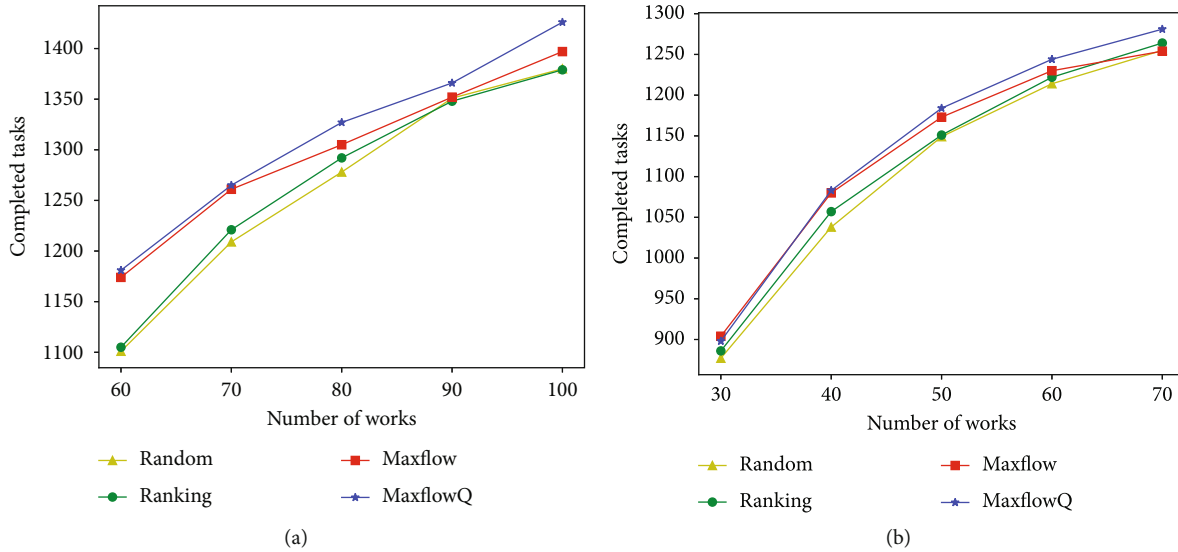


FIGURE 10: Number of tasks completed by different workers: (a) Gowalla and (b) Foursquare.

different experimental environments. The baseline matching strategy in each period is as follows:

- (1) Random: the platform randomly selects tasks for workers to execute in each period
- (2) Ranking algorithm [43]: the ranking algorithm for online bipartite graph matching problem proposed by Karp et al. The workers are sorted before the start of each period, and the platform matches the workers according to the order until the platform reaches the deadline
- (3) Maximum flow [13]: worker task matching is performed from the perspective of maximizing the number of completed tasks in each period. We use the Ford-Fulkerson algorithm to calculate the maximum flow
- (4) Maximum Q value maximum flow (MaxflowQ): strategy of this paper. Based on the maximum flow strategy, considering the impact of current decisions on future decisions, the DDQN model [12] is introduced to train historical data to generate a Q network for real-time prediction and optimize the dynamic matching strategy. In each period, the action state value output by the Q network is used as the weight of the maximum flow matching, and the maximum matching number and the maximum Q value of the platform in the current period are obtained

5.4.1. The Effect of Different Numbers of Workers on Task Assignment. When the platform performs dynamic assignment, multiple workers will influence each other, resulting in changes in the number of tasks completed by workers. This section compares the number of tasks completed by the platform on the Gowalla dataset and the Foursquare dataset under different total worker counts. The maximum

perception radius is set to 2 km, and the test set is the data of 10 days and 15 days after the training set of Gowalla and Foursquare, respectively. It can be seen from Figure 10(a) that our strategy is better than other comparison strategies in the Gowalla dataset, and as the number of workers increases, our strategy shows better performance and can complete a larger number of tasks than other strategies. Figure 10(b) shows that on the Foursquare dataset, when the number of workers is 30, the strategy of this paper is close to the comparison strategy. With the increase of the number of workers, the strategy of this paper is better than other comparison strategies. However, the maximum flow strategy is more likely to fall into a local period optimum due to the increase of the assignment period and the number of workers. Combining Figures 10(a) and 10(b), it can be seen that due to the addition of the Q network in the decision-making process, our strategy shows better performance when the number of workers increases. Furthermore, it is not easy to fall into the optimum of the local period, which reflects the effectiveness of introducing reinforcement learning to consider the impact of current decisions on future decisions.

5.4.2. The Impact of Different Task Test Sets on Task Assignment. In the dynamic assignment process, the environment is dynamic and uncertain. To explore the performance of the framework in different environments, the experiments in this section select test task sets of different days in the Gowalla dataset and the Foursquare dataset to compare the number of tasks completed by the platform. The maximum perception radius of workers is set to 2 km, and the number of test workers on the Gowalla and Foursquare test sets is 70 and 50, respectively. It can be seen from Figure 11(a) that the strategy in this paper is better than other comparison strategies under different test sets in the Gowalla dataset. Moreover, this strategy outperforms the other strategies when the test set is sparse with 7 and 9 days of data. From Figure 11(b), we can see that in the Foursquare

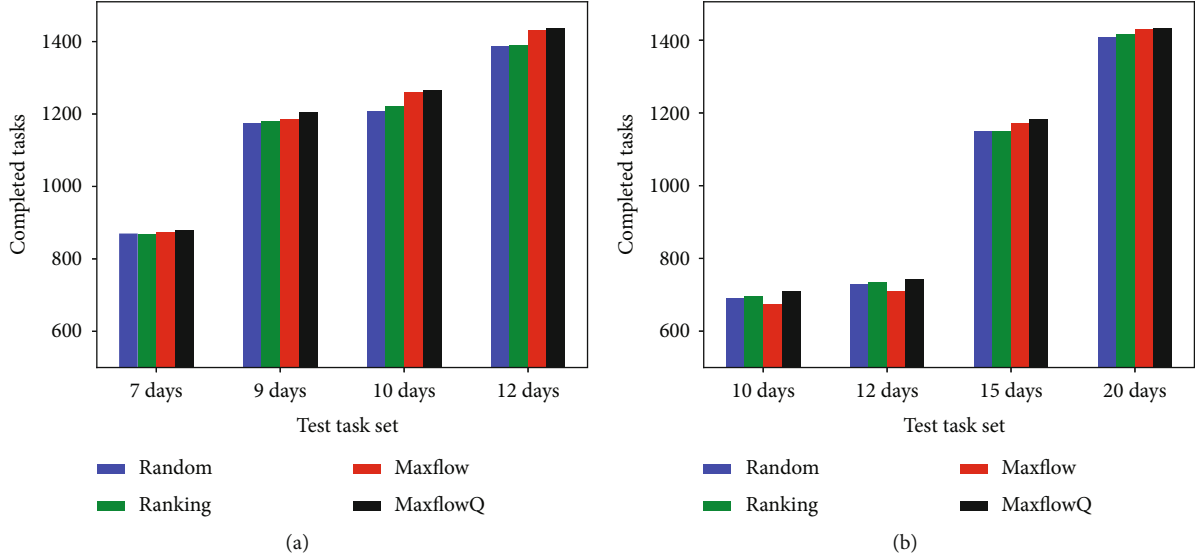


FIGURE 11: Number of tasks completed by workers in different test task sets: (a) Gowalla and (b) Foursquare.

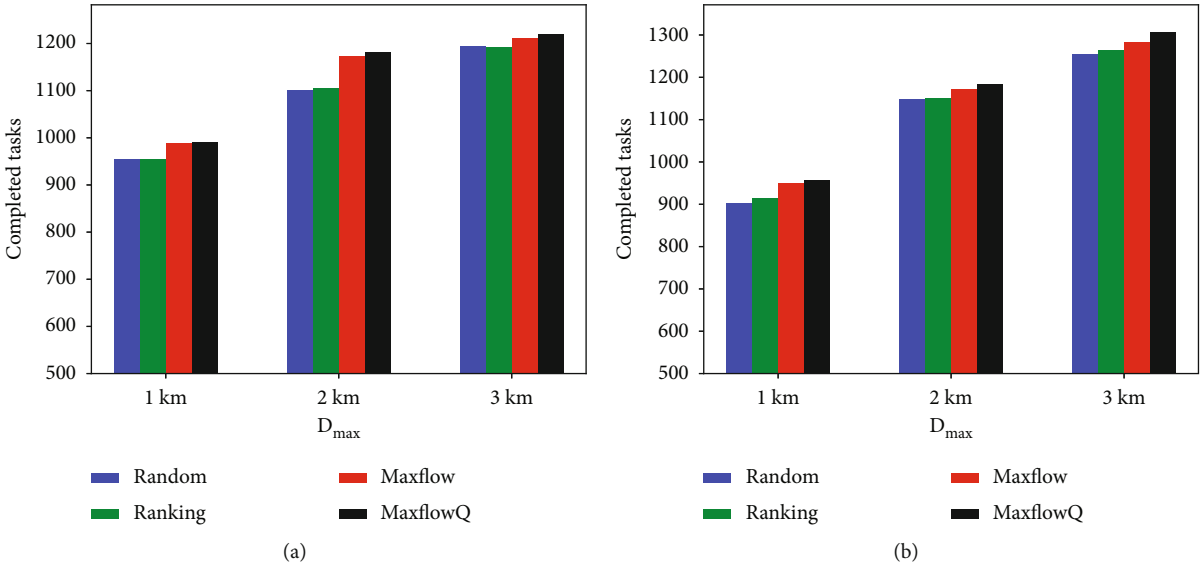


FIGURE 12: Number of tasks completed by workers under different perception radius: (a) Gowalla and (b) Foursquare.

dataset, due to the sparse tasks in the 10 days and 12 days test sets, the maximum flow matching strategy falls into a local period optimum, resulting in a low number of overall task assignments. In the 20-day task set, the number of tasks completed by the maximum flow matching strategy is close to the strategy in this paper due to the denser tasks. The combined Figures 11(a) and 11(b) show that the strategy of this paper outperforms other comparative algorithms under different test sets. And it is not easy to fall into the local period optimum, which indicates that the dynamic assignment framework is stable in the actual test environment.

5.4.3. The Effect of Different Perception Radius D_{max} on Task Assignment. The worker's perception range determines the task set that a worker can choose in each period, which will affect the platform's matching result for the worker. The

experiments in this section compare the number of tasks completed by the platform on the Gowalla dataset and the Foursquare dataset for workers with different maximum perception radius D_{max} . The test set in the Gowalla dataset is 10 days of check-in data, and the number of test workers is 60. The test set in the Foursquare dataset is 15 days of check-in data, and the number of test workers is 50. At the same time, we trained Q networks with maximum perception radius of 1 km and 3 km, respectively, based on the Q network with maximum perception radius of 2 km. From Figures 12(a) and 12(b), it can be seen that the strategy in this paper can complete more tasks under different perception ranges. With the increase of the worker's perception range, the worker can perceive more tasks at each assignment, and the platform can more easily find the optimal assignment for each worker, showing better performance.

6. Conclusion

This paper constructs a dynamic task assignment framework for MCS based on deep reinforcement learning. First, a single worker is modeled for the Markov decision process, and the DDQN model is used to train the historical task set to generate a Q network for real-time prediction. Then, in the process of dynamic task assignment, the maximum Q value maximum flow matching strategy is used to maximize the number of completed tasks in each period while enabling workers to travel to task-intensive areas in order to avoid no feasible tasks in the perceived range in the future. As a result, the global maximum task assignment of the platform is achieved. In the following work, we will focus on the heterogeneity of workers and tasks, explore the cooperation and competition of workers in large-scale task assignments, and try to introduce related theoretical knowledge, such as game theory and multiagent reinforcement learning to optimize the frame.

Data Availability

The experiment uses the datasets of the Gowalla dataset [41] and the Foursquare dataset [42]. All datasets can be accessed from the relevant references.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62172182).

References

- [1] X. Wang, S. Garg, H. Lin, G. Kaddoum, J. Hu, and M. M. Hassan, "Heterogeneous blockchain and AI-driven hierarchical trust evaluation for 5G-enabled intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 2, pp. 2074–2083, 2023.
- [2] N. Maisonneuve, M. Stevens, M. Niessen, and L. Steels, "Noisetube: measuring and mapping noise pollution with mobile phones," in *Information Technologies in Environmental Engineering*, pp. 215–228, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [3] B. Guo, H. Chen, Z. Yu, X. Xie, S. Huangfu, and D. Zhang, "FlierMeet: a mobile crowdsensing system for cross-space public information reposting, tagging, and sharing," *IEEE Transactions on Mobile Computing*, vol. 14, no. 10, pp. 2020–2033, 2015.
- [4] S. Reddy, K. Shilton, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using context annotated mobility profiles to recruit data collectors in participatory sensing," in *Location and Context Awareness*, pp. 52–69, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [5] G. Cardone, L. Foschini, P. Bellavista et al., "Fostering participation in smart cities: a geo-social crowdsensing platform," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 112–119, 2013.
- [6] M. Zhang, P. Yang, C. Tian et al., "Quality-aware sensing coverage in budget-constrained mobile crowdsensing networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7698–7707, 2016.
- [7] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Pervasive Computing: 8th International Conference, Pervasive 2010. Proceedings 8*, pp. 138–155, Helsinki, Finland, 2010.
- [8] L. Kazemi and C. Shahabi, "GeoCrowd: enabling query answering with spatial crowdsourcing," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*, pp. 189–198, New York, NY, USA, November 2012.
- [9] L. Kazemi, C. Shahabi, and L. Chen, "GeoTruCrowd: trustworthy query answering with spatial crowdsourcing," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'13)*, pp. 314–323, New York, NY, USA, November 2013.
- [10] P. Cheng, X. Lian, L. Chen, and C. Shahabi, "Prediction-based task assignment in spatial crowdsourcing," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pp. 997–1008, San Diego, CA, USA, April 2017.
- [11] P. Cheng, X. Lian, Z. Chen et al., "Reliable diversity-based spatial crowdsourcing by moving workers," *Proceedings of the VLDB Endowment*, vol. 8, no. 10, pp. 1022–1033, 2015.
- [12] H. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the thirtieth AAAI conference on artificial intelligence (AAAI'16)*, pp. 2094–2100, 2016, AAAI press.
- [13] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.
- [14] X. Zhang, Z. Yang, Y. Liu, J. Li, and Z. Ming, "Toward efficient mechanisms for mobile crowdsensing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 1760–1771, 2017.
- [15] Y. Gong, C. Zhang, Y. Fang, and J. Sun, "Protecting location privacy for task allocation in ad hoc mobile cloud computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 110–121, 2018.
- [16] Z. Liu, Z. Li, and K. Wu, "UniTask: a unified task assignment design for mobile crowdsourcing-based urban sensing," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6629–6641, 2019.
- [17] M. Xiao, J. Wu, L. Huang, R. Cheng, and Y. Wang, "Online task assignment for crowdsensing in predictable mobile social networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 8, pp. 2306–2320, 2017.
- [18] S. Yang, F. Wu, S. Tang et al., "Selecting most informative contributors with unknown costs for budgeted crowdsensing," in *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pp. 1–6, Beijing, China, June 2016.
- [19] Y. Liu, B. Guo, Y. Wang, W. Wu, Z. Yu, and D. Zhang, "TaskMe: multi-task allocation in mobile crowd sensing," *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16)*, 2016, pp. 403–414, New York, NY, USA, September 2016.
- [20] X. Zhang, Z. Yang, Y.-J. Gong, Y. Liu, and S. Tang, "SpatialRecruiter: maximizing sensing coverage in selecting workers for spatial crowdsourcing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 5229–5240, 2017.

- [21] X. Zhang, Z. Yang, and Y. Liu, "Vehicle-based bi-objective crowdsourcing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3420–3428, 2018.
- [22] M. Li, Y. Gao, M. Wang, C. Guo, and X. Tan, "Multi-objective optimization for multi-task allocation in mobile crowd sensing," *Procedia Computer Science*, vol. 155, pp. 360–368, 2019.
- [23] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*, pp. 703–714, New York, NY, USA, September 2014.
- [24] J. Ji, Y. Guo, D. Gong, and W. Tang, "MOEA/D-based participant selection method for crowdsensing with social awareness," *Applied Soft Computing*, vol. 87, article 105981, 2020.
- [25] Z. Song, C. H. Liu, J. Wu, J. Ma, and W. Wang, "QoI-aware multitask-oriented dynamic participant selection with budget constraints," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4618–4632, 2014.
- [26] J. Wang, Y. Wang, D. Zhang et al., "Fine-grained multitask allocation for participatory sensing with a shared budget," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1395–1405, 2016.
- [27] J. Wang, Y. Wang, D. Zhang, W. Feng, and L. Ma, "PSAllocator: multi-task allocation for participatory sensing with sensing capability constraints," in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17)*, pp. 1139–1151, New York, NY, USA, February 2017.
- [28] H. Shah-Mansouri and V. Wong, "Profit maximization in mobile crowdsourcing: a truthful auction mechanism," in *2015 IEEE International Conference on Communications (ICC)*, pp. 3216–3221, London, UK, June 2015.
- [29] C. Zhou, C.-K. Tham, and M. Motani, "QOATA: Qoi-aware task allocation scheme for mobile crowdsensing under limited budget," in *IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pp. 1–6, 2015.
- [30] M. H. Cheung, F. Hou, J. Huang, and R. Southwell, "Distributed time-sensitive task selection in mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 6, pp. 2172–2185, 2021.
- [31] R. Estrada, R. Mizouni, H. Otok, A. Ouali, and J. Bentahar, "A crowd-sensing framework for allocation of time-constrained and location-based tasks," *IEEE Transactions on Services Computing*, vol. 13, no. 5, pp. 769–785, 2020.
- [32] J. Ji, Y. Guo, X. Gao, D. Gong, and Y. Wang, "Q-learning-based hyperheuristic evolutionary algorithm for dynamic task allocation of crowdsensing," *IEEE Transactions on Cybernetics*, vol. 53, no. 4, pp. 2211–2224, 2023.
- [33] S. Akter, T.-N. Dao, and S. Yoon, "Time-constrained task allocation and worker routing in mobile crowd-sensing using a decomposition technique and deep Q-learning," *IEEE Access*, vol. 9, pp. 95808–95822, 2021.
- [34] X. Wang, M. Peng, H. Lin, Y. Wu, and X. Fan, "A privacy-enhanced multiarea task allocation strategy for Healthcare 4.0," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 3, pp. 2740–2748, 2023.
- [35] X. Tao and W. Song, "Task allocation for mobile crowdsensing with deep reinforcement learning," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–7, Seoul, Korea (South), May 2020.
- [36] X. Wang, S. Garg, H. Lin, G. Kaddoum, J. Hu, and M. S. Hossain, "A secure data aggregation strategy in edge computing and blockchain-empowered Internet of Things," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14237–14246, 2022.
- [37] J. Han, Z. Zhang, and X. Wu, "A real-world-oriented multi-task allocation approach based on multi-agent reinforcement learning in mobile crowd sensing," *Information*, vol. 11, no. 2, p. 101, 2020.
- [38] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [39] Z. Wang, Z. Qin, X. Tang, J. Ye, and H. Zhu, "Deep reinforcement learning with knowledge transfer for online rides order dispatching," in *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 617–626, Singapore, November 2018.
- [40] Z. Ni, H. Liu, X. Zhu, Y. Zhao, and J. Ran, "Task allocation strategy of spatial crowdsourcing based on deep reinforcement learning," *Pattern Recognition and Artificial Intelligence*, vol. 34, no. 3, pp. 191–205, 2021.
- [41] <http://snap.stanford.edu/data/loc-Gowalla.html>.
- [42] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu, "Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 129–142, 2015.
- [43] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pp. 352–358, 1990.