DE-DQN: A Dual-Embedding Based Deep Q-Network for Task Assignment Problem in Spatial Crowdsourcing

Yucen Gao[1], Dejun Kong[1], Haipeng Dai[2], Xiaofeng Gao[1(✉)], Jiaqi Zheng[2], Fan Wu[1], and Guihai Chen[2]

[1] Shanghai Jiao Tong University, Shanghai 200240, China
{guo_ke,kdjkdjkdj99}@sjtu.edu.cn, {gao-xf,fwu}@cs.sjtu.edu.cn
[2] Nanjing University, Nanjing 210008, China
{haipengdai,jzheng,gchen}@nju.edu.cn

Abstract. Along with the rapid development of sharing economy, spatial crowdsourcing has become a hot topic of general interests in recent years. One of its core issues is the task assignment problem, in which tasks are released on the platforms and then assigned to available workers. Due to the various characteristics of tasks and workers, finding the optimal assignment scheme and routing plan are difficult.

In this paper, we define the utility-driven destination-aware spatial task assignment problem (UDSTA), which is proved to be NP-complete. Our goal is to maximize the total utility of workers. We propose a dual-embedding based deep Q-Network (DE-DQN) to sequentially assign tasks to suitable workers. Specifically, we design a utility embedding to reflect the top-k utility tasks for workers and worker-task pairs, and propose a coverage embedding to represent the potential future utility of an assignment action. For the first time, we combine the dual embedding with DQN to realize the multi-task and multi-worker matching, and obtain route plans of workers. Experiments based on both synthetic and real-world datasets indicate that DE-DQN performs well and shows significant advantages over the baseline methods.

Keywords: Dual embedding · Deep Q-Network · Task assignment · Spatial crowdsourcing

1 Introduction

Spatial crowdsourcing is an emergent working mode in recent years, which decomposes sophisticated tasks from task publishers into multiple small and easy tasks, and then assigns these tasks to numerous mobile workers with various backgrounds to finish the tasks efficiently and fast. Along with the rapid

This work was supported by National Key R&D Program of China [2019 YFB2102200]; National Natural Science Foundation of China [61872238, 6187 2178, 61972254], Shanghai Municipal Science and Technology Major Project [2021 SHZDZX0102] and Huawei Cloud [TC20201127009]. Thanks for the help of Wei Liu.

B. Li et al. (Eds.): APWeb-WAIM 2022, LNCS 13421, pp. 280–295, 2023.
https://doi.org/10.1007/978-3-031-25158-0_22

development of mobile intelligent devices, spatial crowdsourcing is extending to many aspects of the society. Many related applications based on spatial crowdsourcing systems are proposed and employed in practice such as Ear-phone [14] for urban noise sensing and Uber [17] for taxis hailing.

Traditional spatial crowdsourcing problems concentrate on task assignment, quality control, incentive system, and privacy protection. Nevertheless, they typically overlook mobility, online situation and tempo-spatial characteristics, which are crucial to many applications. Besides, these features also make the problems more complicated. Therefore, researchers have paid much attention to explore easier, quicker and more robust methods to deal with spatial crowdsourcing.

One of the core issues in spatial crowdsourcing is the task assignment. Task assignment is to allocate tasks with unique attributes such as revenue and location to workers in service who also have specific characteristics such as service scope and ability. Such a process of task-worker matching is realized via a spatial crowdsourcing platform as shown in Fig. 1. The task publishers submit tasks to be completed with various requirements to the platform. The workers join in the platform through recruitment. Then, the platform matches compatible tasks with workers according to the tasks pool and workers pool. Nevertheless, if the scales of tasks and workers increase, the computational complexity of the problem may increase in factorial order. Thus, proposing efficient and fast methods for the situations of large-scale data deserves in-depth research.

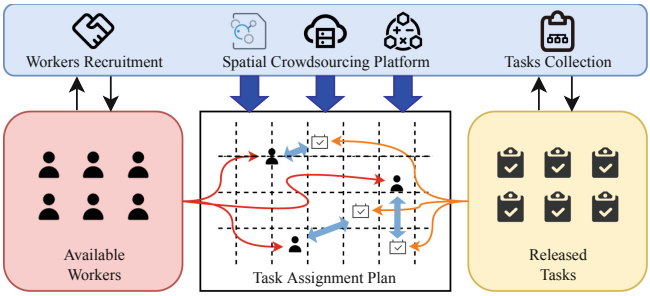


Fig. 1. Illustration of task assignment in spatial crowdsourcing.

In our scenario, we consider the problem of which multiple tasks and workers are distributed in a spatial area. A worker can receive and finish a limited number of tasks, while a task can only be assigned to one worker. A cost emerges when a worker deals with a task, which is proportional to the distance between the worker and the task. When a task is completed, the worker will receive the corresponding reward. The goal of the problem is to maximize the total utility of all the workers given the cost and reward. The problem is named as utility-driven destination-aware spatial task assignment problem (UDSTA). Apparently, there may be conflicts between global optimum and local optimum (*e.g.*, high-reward tasks are preferred by many workers). From the perspective of workers,

maximizing their own revenue can be a naive strategy for task selection, which is mainly associated to less working interval and larger task reward. Less working interval leads to preferring neighboring tasks, while larger task revenue leads to preferring tasks with high reward. There may exist a decision dilemma when the two strategies conflict with each other, making the optimization more difficult.

As for UDSTA, traditional algorithms like greedy algorithms and heuristic algorithms may fall into local optimum and cannot obtain optimal solutions. Even worse, when the scale of the problem is large, solving the problem is very time-consuming. Hence, to leverage the advantages of machine learning techniques in finding global optimal solutions and addressing large-scale problems, we propose a new value based deep learning framework called dual-embedding based deep Q-Network (DE-DQN), which innovatively utilizes dual embedding vectors as the input of DQN to learn the optimal strategy of task assignment. DE-DQN can significantly improve the efficiency of problem solving and performs best in large-scale scenarios. What's more, since UDSTA problem is a generic setting for the task assignment problem, the proposed DE-DQN can be easily adapted to address many variant problems such as sweep coverage and vehicle dispatching by adding time-related factors and modifying the optimization goal.

2 Problem Statement and Preliminaries

In the paper, we pay attention to the scenarios such as region monitoring, where location distribution of tasks and workers has impact on solutions and time-related constraints are less important. In the scenario, tasks are scattered in different locations. To obtain profit, workers are motivated to complete some tasks. Simultaneously, workers would cover the costs such as travel costs when completing tasks. Hence, the utility for a worker is equal to profit minus cost and the objective is to maximize the overall utility for all workers. However, time-related factors can be linked to the feasible assignment and thus included in the DQN framework described below in scenarios where time constraints need to be considered, which we plan to investigate in more detail in future work.

Following most existing works in crowdsourcing [22] , we adopt the round-based model. In each round, the platform operates on a set of pended tasks and workers. Basic definitions related to worker, task and utility are introduced here.

Definition 1 (Worker). *A worker w_i is denoted by a tuple $\langle l_i^w, r_i, C_i \rangle$, where l_i^w is the current location of worker w_i; r_i is w_i's route, which consists of tasks assigned to w_i; C_i is w_i's capacity limit of workload.*

Definition 2 (Spatial task). *A spatial task s_j is denoted by $\langle l_j^s, p_j \rangle$, where l_j^s is the geographical location of task s_j; p_j is the profit when task s_j is completed.*

Definition 3 (Worker's utility for accepting task). *The utility of worker w_i to accept task s_j is defined by a function $u(\cdot)$ where*

$$u(w_i, s_j) = p_j - cost(w_i, s_j) \tag{1}$$

Here $cost(w_i, s_j)$ represents the cost for w_i to complete s_j.

Definition 4 (Overall utility). *The overall utility is defined as the sum of workers' utilities, i.e.,*

$$U_{all} = \sum_{s_j^*} p_j - \sum_{w_i} cost(w_i) \tag{2}$$

where s_j^* represents the assigned task and $cost(w_i) = \sum_{s_j \in r_i} cost(w_i, s_j)$ represents the overall cost of worker w_i.

We consider the task assignment mode with several constraints as follows:

- **Uniqueness constraint:** A spatial task can only be assigned to one worker.
- **Spatial constraint:** A spatial task can be completed only if the worker arrives at the location of the task.
- **Capacity constraint:** The number of tasks assigned to w_i cannot exceed the workload capacity limit C_i.

Without loss of generality, we also have the following assumptions.

Assumption 1. *Since a worker should arrive at the location of a task to complete it, the $cost(w_i, s_j)$ may be proportional to $dist(l_i^w, l_j^s)$, which means*

$$cost(w_i, s_j) = \beta_i \cdot dist(l_i^w, l_j^s) \tag{3}$$

where β_i is the cost per unit travel distance for w_i and $dist(\cdot)$ is distance function.

Assumption 2. *According to individual rationality assumption, a worker accepts the task only if the utility is positive, i.e., $u(w_i, s_j) > 0$.*

Based on the aforementioned discussions, we formulate the *Utility-driven Destination-aware Spatial Task Assignment* problem (UDSTA) as follows.

Definition 5 (Utility-driven Destination-aware Spatial Task Assignment problem). *Given a worker set $\mathcal{W}$ and a task set $\mathcal{S}$, the Utility-driven Destination-aware Spatial Task Assignment problem (UDSTA) aims to find a global assignment solution $\mathbb{A}$ to maximize the overall utility for workers, while satisfying the aforementioned assumptions and constraints.*

Figure 2 shows a toy example of the UDSTA problem with the positions of 2 workers and 9 tasks, where the radius of nodes represents the profit of tasks, and the length of edges represents the distance between two locations. Assume the two workers' cost per unit travel distance β_1 and β_2 are both equal to 1, the proper task sequence for worker w_1 with capacity $c_1 = 3$ will be $r_1 = [s_2, s_3, s_5]$. In fact, worker w_1 chooses the task with the highest utility at each step. However, such greedy idea does not always make sense. If worker w_2 with capacity $c_2 = 2$ adopts a greedy policy, he/she will pick up s_6 instead of s_8 at the first step. Hence, the greedy route r_2' will be $[s_6, s_7]$, obtaining a total utility of 6. However, a better route $r_2 = [s_8, s_9]$ obtains a utility of 7, since there exist better neighboring tasks around s_8 compared with s_6. Our proposed DE-DQN method will focus on the important potential future utility.

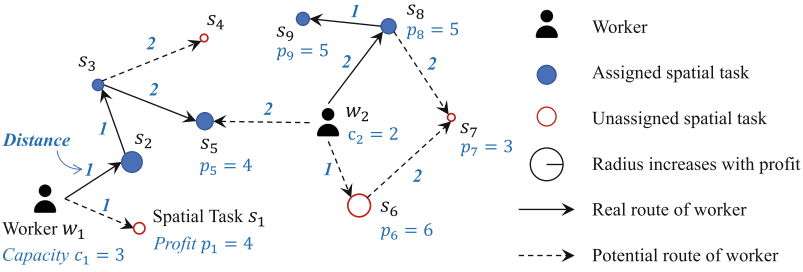


Fig. 2. A toy example with 2 workers and 9 spatial tasks.

NP-completeness of the UDSTA Problem. We prove the NP-completeness of the UDSTA problem with a reduction from a variation of the TSP problem, say, Traveling Salesman Path problem (abbreviated as TSP-Path) without returning to the original source, which has been proved to be NP-complete [6].

Theorem 1. *The UDSTA problem is NP-Complete.*

Proof. It is trivial to check a certificate of the UDSTA problem in polynomial time. Thus the UDSTA is an NP problem. Next, let us consider a reduction from TSP-Path. Given $G(V, E)$ with $|V| = n$, we define $cost(e) = 1, \forall e \in E$. Assume that $n - 1$ nodes represent tasks whose profits are 2, while the remaining node represents the worker with capacity $c = n - 1$. To maximize the overall utility, the worker will accept all the $n - 1$ tasks. We add edges E' to G to make G a complete graph and define $cost(e) = 2, \forall e \in E'$. In this situation, maximizing overall utility means minimizing cost. We consider the decision version of the UDSTA problem, which is "Is there a route of cost at most $n-1$?". Now we can use Cook's reduction and solve the decision version of the TSP-Path problem by an oracle of UDSTA. Therefore, UDSTA is NP-Complete. □

The important notations in the rest of the paper are summarized in Table 1.

3 DE-DQN: Dual-Embedding Based Deep Q-Network for Task Assignment

Since cost is assumed to be proportional to distance between worker and task, we utilize the graph to model the destination-aware spatial crowdsourcing and propose a neural network-based learning method on the graph to solve the UDSTA problem. We first describe the dual-embedding approach to obtain the vector representations of workers and tasks for flexible computation in the neural network-based learning. Then, we use a Deep Q-Network (DQN) to sequentially assign tasks to proper workers and obtain the global task assignment solution.

Table 1. Definitions and notations

Symbol	Definition
$\mathcal{W}$	worker set, each worker $w_i \in \mathcal{W}$
$\mathcal{S}$	spatial task set, each task $s_j \in \mathcal{S}$
$\mathcal{S}^*$	assigned spatial task set
w_i	a worker $w_i = \langle l_i^w, r_i, C_i \rangle$
s_j	a spatial task $s_j = \langle l_j^s, p_j \rangle$
s_j^*	tasks received by workers
l_i^w	location of worker w_i
r_i	task sequence received by worker w_i
C_i	limit of task received by worker w_i
l_j^s	location of task s_j
p_j	profit when task s_j is completed
$dist(\cdot)$	distance between two locations
β_i	cost per unit travel distance for w_i
$cost(w_i, s_j)$	cost for worker w_i to complete task s_j
$u(w_i, s_j)$	utility for worker w_i to complete task s_j
U_{all}	overall utility for worker set, $U_{all} = \sum_{s_j^*} p_j - \sum_{s_j^*} cost(w_i, r_i, s_j)$

3.1 Dual-Embedding

The crowdsourcing graph is composed of a large number of nodes with worker and task features, and edges with distance information, which makes the graph complex and difficult to directly perform neural network-based learning on it. As is shown in Fig. 2, reasonable consideration of potential future utility makes sense. Since the potential future utility of an assignment can be measured from the breadth and depth perspective, we propose a dual-embedding method for nodes, using the utility embedding to reflect the top-k neighboring tasks' utilities from the breadth perspective and the coverage embedding to represent a worker's potential future utility linked with the remaining capacity from the depth perspective.

Utility Embedding: According to a worker's historical record, current location, and neighboring uncompleted tasks, we construct the utility embedding vector to reflect the top-k neighboring tasks' utilities for the worker, which is helpful to provide a larger search space. To consider the potential utility of a longer route, we similarly construct the utility embedding vector for pair (w_j, s_j) to reflect the top-k neighboring utilities when s_j is assigned to w_i.

Representation Vector for Worker: For worker w_i, we use w_i^k to represent the index of the k_{th} highest utility task for w_i, and then obtain uncompleted task sequence with the top-k highest utilities for w_i: $[s_{w_i^1}, s_{w_i^2}, \cdots, s_{w_i^k}]$. Therefore, the utility embedding for w_i is

$$\mathbf{V}_{w_i}^u = (u(w_i, s_{w_i^1}), u(w_i, s_{w_i^2}), \cdots, u(w_i, s_{w_i^k})) \tag{4}$$

Representation Vector for Pair: An assignment can be seen as a worker-task pair (w_j, s_j), which means that s_j is assigned to w_i and w_i arrives at l_j^s. In the scenario, we define $u_{w_i}(s_j, s_k) = p_k - \beta_i \cdot dist(l_j^s, l_w^s)$ to represent the utility for w_i to complete s_k after completing s_j, and use $(w_i s_j)^k$ to represent the index of the k_{th} highest utility task when w_i arrives at l_j^s. Hence, the utility embedding for pair (w_j, s_j) is

$$\mathbf{V}^u_{w_i s_j} = (u_{w_i}(s_j, s_{(w_i s_j)^1}), u_{w_i}(s_j, s_{(w_i s_j)^2}), \cdots, u_{w_i}(s_j, s_{(w_i s_j)^k})) \tag{5}$$

Coverage Embedding: The coverage embedding reflects the potential future utility for an assignment considering the remaining workload capacity of a worker. Specifically, for pair (w_i, s_j), the potential future utility can be denoted by the sum of the highest neighboring task utilities in the propagation chain with the remaining workload capacity rc_i as length. We define $s^m_{(w_i s_j)^1}$ to represent the m_{th} highest utility task in the propagation chain. For instance, $s^2_{(w_j s_j)^1} = s_{(w_i s_{(w_i s_j)^1})^1}$. Especially, $s^0_{(w_i s_j)^1} = s_j$. Hence, for pair (w_i, s_j), we can express the potential future utility of assigning $s^k_{w_i s_j}$ to w_i as

$$u^c_{w_i}(s_j, s_{(w_i s_j)^k}) = u_{w_i}(s_j, s_{(w_i s_j)^k}) + \sum_{m=1}^{rc_i - 1} u_{w_i}(s^{m-1}_{(w_i s_j)^1}, s^m_{(w_i s_j)^1}) \tag{6}$$

Therefore, for pair (w_i, s_j), we concatenate the top-k potential future utilities to form the coverage embedding $\mathbf{V}^c(w_i, s_j)$ as

$$\mathbf{V}^c(w_i, s_j) = (u^c_{w_i}(s_j, s_{(w_i s_j)^1}), \cdots, u^c_{w_i}(s_j, s_{(w_i s_j)^k})) \tag{7}$$

3.2 Framework and Training for DE-DQN

Reinforcement learning is a learning process of studying proper interaction with the environment to maximize a special reward. To solve the UDSTA problem, we adopt the DQN, which is a structure combining Q-learning with the deep neural network, to assign tasks to proper workers. Specifically, the framework of our DE-DQN is shown in Fig. 3.

In the DE-DQN framework, the agents are the decision makers. The environment consists of the worker and task graph and determined task sequences. The agent takes an action to assign a task to the proper worker according to the environment. The environment then gives a reward as feedback to the agent. The goal of the agent is to maximize the long-term gain. These components are described as follows.

State: The state reflects the current task assignment situation in the spatial crowdsourcing network. The state vector concatenates three parts. The first part is the mean of the top-k utilities for all workers $\mathbf{V}^u_W = \frac{1}{|W|} \sum_{w_i \in W} \mathbf{V}^u_{w_i}$. The second part indicates the total remaining workload capacity $rc = \sum_{w_i \in W} rc_i$. The third part records the sum of cumulative utility and cost of completed tasks $uc = (\sum_{s_j \in \mathcal{S}^*} p_j, \sum_{w_i} cost(w_i))$. The state vector $x_s = (\mathbf{V}^u_W || rc || uc)$.

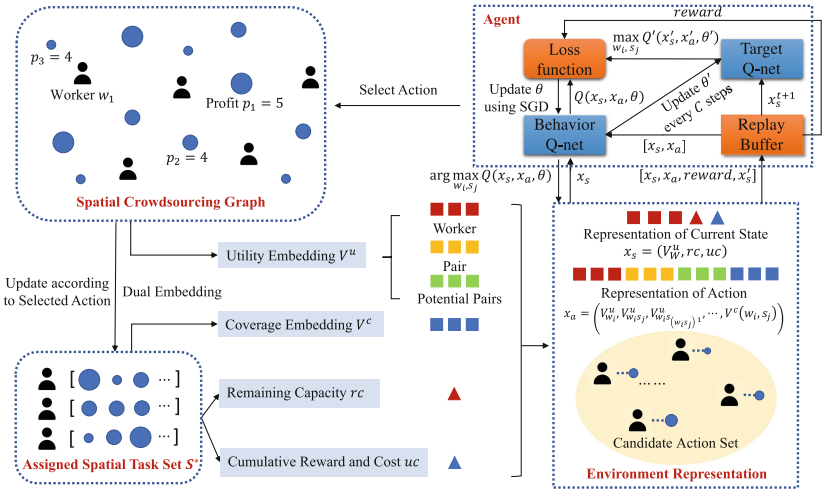


Fig. 3. The illustration of DE-DQN. DE-DQN first calculates the utility embedding $\mathbf{V}^u$ and coverage embedding $\mathbf{V}^c$ for workers and pairs. The state representation x_s is the concatenation of three parts: mean of the utility embedding for workers $\mathbf{V}^u_W$, remaining capacity rc, and cumulative reward and cost uc. The assignment action representation x_a is the concatenation of utility embedding and coverage embedding. DE-DQN selects the proper action based on the current state and candidate action set. After determining a worker-task pair, the agent receives the reward, and then updates the corresponding variables and representations.

Action: The action is to assign a task to a worker. To take the future reward into account, we consider the current task to be assigned and the potential future tasks. Specifically, we concatenate the utility embedding of worker w_i, task s_j, potential future tasks $\{s_{(w_i s_j)^1}, \cdots, s_{(w_i s_j)^k}\}$, and coverage embedding to indicate the action of agent $x_a = (\mathbf{V}^u_{w_i} || \mathbf{V}^u_{w_i s_j} || \mathbf{V}^u_{w_i s_{(w_i s_j)^1}} || \cdots || \mathbf{V}^c(w_i, s_j))$.

Reward: The reward represents the utility obtained. When the agent assigns task s_j to worker w_i, the reward is equal to $p_j - cost(w_i, s_j)$.

Policy: The deep Q-learning is an off-policy algorithm where we employ a neural network to approximate the Q value. Specifically, we use the ϵ-greedy algorithm in the training stage to do more exploration, and the pure greedy algorithm in the testing stage to maximize the overall reward.

The ϵ-greedy policy in the training stage is

$$\gamma_{\epsilon-greedy}(x_s) = \begin{cases} \underset{((w_i,s_j)|s_j \in \mathcal{S}-\mathcal{S}^*)}{\arg\max} Q & \text{w.p. } \epsilon \\ \text{a random feasible pair } ((w_i, s_j)|s_j \in \mathcal{S} - \mathcal{S}^*) & \text{o.w.} \end{cases} \tag{8}$$

The pure greedy policy in the testing stage is

$$\gamma_{greedy}(x_s) = \underset{((w_i,s_j)|s_j \in \mathcal{S}-\mathcal{S}^*)}{\arg\max} Q \tag{9}$$

The DE-DQN framework consists of a behavior Q-network and a target Q-network to further improve the stability of Q-value in the training procedure. The two Q-networks both use the Q-value to evaluate the value of assigning a task to a worker in a given state. Specifically, the behavior Q-network is responsible for predicting the value of action in current state, while the target Q-network is responsible for evaluating the expected gain in next state.

The network structures of the two Q-networks are the same, while parameters are different. Let θ and θ' be the parameters of the behavior Q-network and the target Q-network, respectively. The inputs of the neural network are the state x_s and the action x_a. Hence, the behavior Q-network and the target Q-network can be denoted by $Q(x_s, x_a, \theta)$ and $Q'(x'_s, x'_a, \theta')$, respectively. The training objective can be formulated as minimizing the loss function $L(\theta)$:

$$L(\theta) = \mathbf{E}[(\hat{y^t} - Q(x_s^t, x_a^t, \theta))] \tag{10}$$

$$\hat{y^t} = reward^t + \eta \max_{(w_i, s_j)} Q'(x_s^{t+1}, x_a^{t+1}, \theta') \tag{11}$$

where x_s^t is the state vector at step t, x_a^t is the action vector at step t, and $\eta \in [0, 1]$ is the discount factor.

To reduce the correlation between the predicted Q-value and the target Q-value, the parameters of the target Q-network remains unchanged in C training steps. After C steps, the parameters of the behavior Q-network are copied to update θ' in the target Q-network.

Algorithm 1: DE-DQN Agent Training

Input: Spatial crowdsourcing graph, worker set $\mathcal{W}$, task set $\mathcal{S}$.
Output: Parameters of Q-network.

Initialize replay buffer D;
Initialize parameters of the behavior and the target Q-networks, $\theta = \theta'$;
for *episode* $= 1$ *to* E **do**
  Initialize the assigned spatial task set $\mathcal{S}^* = \emptyset$;
  for $t = 1$ *to* $\sum_{w_i} C_i$ **do**
    Select action by $\gamma_{\epsilon-greedy}$ policy;
    Calculate $reward^t$;
    $x_s^{t+1} \leftarrow f_t(x_s^t, x_a^t)$;
    Store $[x_s^t, x_a^t, reward^t, x_s^{t+1}]$ into D;
    Randomly sample a minibatch data from D;
    if $t = k$ **then**
      $y^t = reward^t$;
    else
      $y^t = reward^t + \eta \max_{(w_i, s_j)} Q'(x_s^{t+1}, x_a^{t+1}, \theta')$;
    Use SGD to take a gradient descent step on $(y^t - Q(x_s^t, x_a^t, \theta))^2$;
    After C steps, update $\theta' = \theta$.

The training process for agent is shown in Algorithm 1. We employ the replay buffer proposed in [12] to reduce the dependency among data. For each training episode, we initialize the assigned spatial task set in Line 4. The episode ends when the number of assigned tasks reaches $\sum_{w_i} C_i$, the sum of workload capacity limit of all workers. In each episode, we adopt $\gamma_{\epsilon-greedy}$ policy to choose an action from candidate sets. After selecting an action, we update the representation of state vector through a state transition function in Line 8. After obtaining the reward and the next state, we save it in the replay buffer. We update the parameters in Q-network by stochastic gradient descent (SGD) method. After C steps, we update the parameters in the target Q-network.

4 Experiments

In this section, we conduct experiments on both synthetic and real-world datasets and evaluate performances of baseline algorithms and our DE-DQN framework.

4.1 Experiment Setups

Dataset Overview: The experiments are carried out on two datasets: Gowalla[1] and synthetic (SYN). Gowalla is an open source real-world dataset [2], which includes a total of $6,442,890$ check-ins of users over the period of Feb 2009-Oct 2010. We choose the locations where the number of user check-ins is greater than 100 as the initial locations of workers, and the locations where the number of item check-ins is greater than 10 as the locations of tasks by random sampling in the longitude range from -120 to -110 and the latitude range from 30 to 40. The task profit is determined by sampling the order value from Didi Chuxing Chengdu dataset. Table 2 shows the percentage of tasks in different value ranges, which indicates that the number of tasks with high profits is often small in reality. Besides, a distance conversion function is used to calculate the distance between points with longitude and latitude. The worker cost per unit travel distance is obtained from a Poisson distribution with the mean of 0.5.

Table 2. Percentage of Tasks in Different Value Ranges

Range	$(0, 0.1]$	$(0.1, 0.2]$	$(0.2, 0.3]$	$(0.3, 0.4]$	$(0.4, 0.5]$	$(0.5, 0.6]$	$(0.6, 0.7]$	$(0.7, 0.8]$	$(0.8, 0.9]$	$(0.9, 1.0]$
Ratio	0.75	0.21	$3.5e^{-2}$	$6.3e^{-3}$	$1.3e^{-3}$	$4.5e^{-4}$	$2.0e^{-4}$	$3.8e^{-5}$	$2.9e^{-5}$	$1.9e^{-5}$

To obtain the synthetic dataset, we generate workers and tasks following a uniform distribution within the $2D$ space $[0, 400]^2$. The task profit is obtained from a probability density function of step descent, which is in line with the above observation in reality. Besides, the worker cost per unit travel distance is obtained from a Poisson distribution with the mean of 0.1.

[1] http://snap.stanford.edu/data/loc-Gowalla.html.

Parameter Settings: The parameter settings are shown in Table 3, where the default ones are marked in bold. As for training the DE-DQN, we do the training with 500 episodes on randomly sampled data from Gowalla and SYN.

Table 3. Parameter setting

Parameter	Description	Value
$\|\mathcal{W}\|$	Worker Set	$50, \mathbf{60}, 70, 80$
$\|\mathcal{S}\|$	Spatial Task Set	$800, 900, \mathbf{1000}, 1100$
Capacity Mean	Mean of $\{C_i\}$	$5, 7, \mathbf{10}, 15$

Baseline methods: We take the following baselines for performance evaluation.

- **DisGreedy:** The DisGreedy algorithm is proposed in [20], which takes distance as the highest priority criterion for selecting the next task. It assigns the nearest task to workers.
- **PftGreedy:** The PftGreedy algorithm is also proposed in [20], which takes profit as the highest priority criterion for selecting the next task. It assigns the task with the highest profit to the nearest worker.
- **Utility Priority:** [4] proposes the Utility Priority algorithm, matching the largest utility worker-task pair at each step.

4.2 Performance Analysis

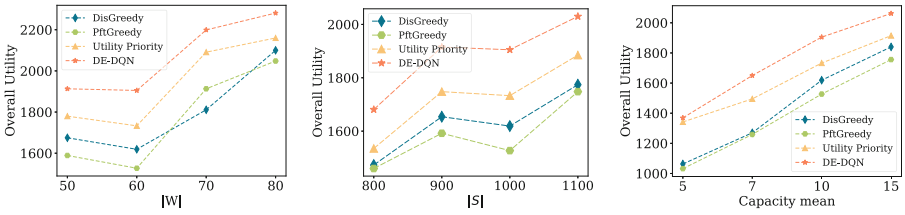
All the experiments are implemented on an Apple M1 chip with 8-core CPU and 16GB unified memory. The DE-DQN and baseline algorithms are implemented with Python 3.6. Tensorflow 2.0 is used to build the machine learning framework. As shown in Fig. 4 and 5, DisGreedy works better than PftGreedy when travel cost is high on Gowalla, while PftGreedy works better than DisGreedy when travel cost is low on SYN. However, DE-DQN and Utility Priority perform well regardless of the change of travel cost.

Effect of $|\mathcal{W}|$: As shown in Fig. 4(a) and 5(a), the performance of DE-DQN is better than that of other baseline algorithms with various $|\mathcal{W}|$. Simultaneously, the performance gap of DE-DQN over Utility Priority increases as $|\mathcal{W}|$ increases, indicating that DE-DQN is suitable for the complex spatial crowdsourcing scenario where there exists a great quantity of workers. We note that the performance of algorithms degrades when $|\mathcal{W}| = 60$ compared with $|\mathcal{W}| = 50$ on Gowalla, indicating that the performance is influenced by data instances.

Effect of $|\mathcal{S}|$: As shown in Fig. 4(b) and 5(b), DE-DQN achieves the highest overall utility with various $\mathcal{S}$. We note that DE-DQN and Utility Priority can discover better tasks as $|\mathcal{S}|$ increases compared with DisGreedy and PftGreedy.

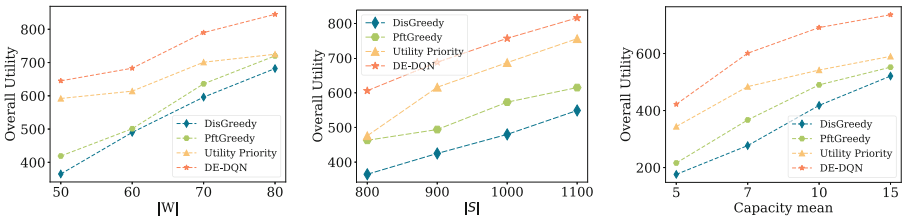
Simultaneously, the performance gap between DE-DQN and Utility Priority decreases as $|\mathcal{S}|$ increases on SYN. This may be because Utility Priority can benefit more from the increase of highly profitable tasks when the total worker workload capacity is constant within a uniform distribution.

Effect of Capacity Mean: As shown in Fig. 4(c) and 5(c), the higher the capacity mean of workers, the better the DE-DQN outperforms the baseline methods when the number of tasks is constant, which indicates that considering future utility is more important as the total worker capacity is closer to $|\mathcal{S}|$.



(a) $|\mathcal{W}|$ (b) $|\mathcal{S}|$ (c) Capacity mean

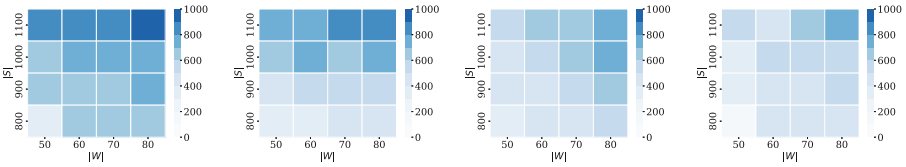
Fig. 4. Comparison with various $|\mathcal{W}|$, $|\mathcal{S}|$, and capacity mean on Gowalla.



(a) $|\mathcal{W}|$ (b) $|\mathcal{S}|$ (c) Capacity mean

Fig. 5. Comparison with various $|\mathcal{W}|$, $|\mathcal{S}|$, and capacity mean on SYN.

Effect of $|\mathcal{W}| \times |\mathcal{S}|$: As shown in Fig. 6, the overall utility grows with the number of workers and the number of tasks for all the four algorithms. DE-DQN outperforms the baseline algorithms under most cases. When there are a relatively small number of tasks and the total workload capacity of workers is small, the performance gap between DE-DQN and other baseline algorithms is not large. However, when there are many tasks and the total workload capacity of workers is close to the task number, the performance gap is large.



(a) DE-DQN (b) Utility priority (c) PftGreedy (d) DisGreedy

Fig. 6. Comparison with the overall utility w.r.t. $|\mathcal{W}| \times |\mathcal{S}|$ on SYN.

In summary, DE-DQN performs well on both synthetic and real-world datasets. What's more, DE-DQN is more suitable for complex spatial crowdsourcing scenario where the total workload capacity of workers is close to the total number of tasks. In this situation, the performance of the DE-DQN can outperform the other comparison algorithms by 20%.

Ablation Study: We conduct experiments with only utility embedding $\mathbf{V}^u$ or coverage embedding $\mathbf{V}^c$ and make comparisons with DE-DQN to study the effect of the dual-embedding. Table 4 shows the overall utility based on different techniques with 1000 and 1200 tasks. The results are consistent with the results shown in Fig. 4(c), which indicates that the potential future utility reflected by coverage embedding $\mathbf{V}^c$ is more important when the total worker workload capacity is closer to the number of tasks. Simultaneously, the results of the ablation study demonstrate that the combination of utility embedding and coverage embedding is beneficial to the overall performance.

Table 4. Comparison with utilities for 60 workers with capacity mean of 10 on SYN

Number of tasks	$\mathbf{V}^u$+DQN	$\mathbf{V}^c$+DQN	DE-DQN
1000	639.70	696.97	762.22
1200	836.53	804.32	928.67

5 Related Work

Task Assignment Problem: Task assignment problem in crowdsourcing has been extensively paid attention to and researched in the past ten years as most task-worker matching requirements in real world application can be formulated as this problem. Task assignment problem can be classified into offline problem or online problem according to whether all the information of tasks and workers is known before the assignment decision process starts [3]. As for the online task assignment, it can be classified into one side online problem [5] and both sides online problem [21]. A common model employed for solving task assignment problem is bipartite graph [11], in which tasks and workers are abstracted into

nodes of both sides. [9] concentrates on ride-sharing routing problem associated to customer selection and proposes a two-stage structure to obtain the optimal solution with pseudo-polynomial time complexity. [4] defines a team-oriented task planning problem, which is actually a multi-task multi-worker matching problem, considering the worker's skill constraint of tasks.

Multiple methods based on different technologies have been studied and mentioned in recent works. [1] explores distributed and centralized algorithms for task selection to improve the quality of task accomplishment, where the distributed one is a game theory based approximation algorithm, the centralized one is a greedy based approximation algorithm. Similarly, [13] respectively define dependency-aware spatial crowdsourcing and cooperation-aware spatial crowdsourcing, and also propose game theory and greedy based approximation algorithms aiming at maximizing the assigned tasks and the cooperation quality separately. Besides traditional methods, [10] first presents a deep reinforcement learning framework on vehicular crowdsourcing problem, both maximizing task accomplishment and coverage fairness, and minimizing the energy cost.

Deep Reinforcement Learning: Reinforcement learning (RL) has become a hotspot in research field recently. Some variants of RL also emerged combined with other methods. Deep reinforcement learning (DRL) combines deep learning methods with traditional RL methods, which is found to be suitable for solving decision optimization problems [16], and therefore has been widely researched for improvement and application. Similar to RL, DRL can also be distinguished by whether it is model-based. As for model-free DRL, Deep Q-Network (DQN) is a popular method based on value function [12], which utilizes convolutional neural network and Q-learning [7], and performs well in many applications including task arrangement in crowdsourcing [8,18]. What's more, [15] proposes auxiliary-task based DRL to achieve multi-objective optimization in participant selection problem of mobile crowdsourcing. [19] introduces geographic partition into spatial crowdsourcing and then conducts the RL method to solve the problem.

6 Conclusion

In this paper, we focus on solving the global task assignment problem in the spatial crowdsourcing. Our goal is to maximize the overall utility for all workers, which is formulated as the utility-driven destination aware spatial task assignment problem and is proved to be NP-complete. To deal with the problem, we propose a DE-DQN framework which balances the current and future utility through the dual-embedding. We construct the proper representations of state, action, and reward of DQN according to our scenario, and use the experience replay buffer to train the Deep Q-Network. The experiments based on both synthetic and real-world dataset verify the effectiveness of the DE-DQN framework. In the future, we will consider the fairness problem where a homogeneity metric is presented to balance the utility among workers.

References

1. Cheung, M.H., Hou, F., Huang, J., Southwell, R.: Distributed time-sensitive task selection in mobile crowdsensing. IEEE Trans. Mob. Comput. **20**(6), 2172–2185 (2021)
2. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1082–1090 (2011)
3. Du, Y., Sun, Y.E., Huang, H., Huang, L., Xu, H., Wu, X.: Quality-aware online task assignment mechanisms using latent topic model. Theoret. Comput. Sci. **803**, 130–143 (2020)
4. Gao, D., Tong, Y., Ji, Y., Xu, K.: Team-oriented task planning in spatial crowdsourcing. In: Asia-Pacific Web and Web-Age Information Management Joint Conference on Web and Big Data, pp. 41–56 (2017)
5. Gao, G., Wu, J., Xiao, M., Chen, G.: Combinatorial multi-armed bandit based unknown worker recruitment in heterogeneous crowdsensing. In: IEEE Conference on Computer Communications, pp. 179–188 (2020)
6. Gutin, G., Punnen, A.P. (eds.): The traveling salesman problem and its variations. Springer Science & Business Media, LLC, Springer (2007). https://doi.org/10.1007/b101971
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25 (2012)
8. Li, B., Zhang, A., Chen, W., Yin, H., Cai, K.: Active cross-query learning: a reliable labeling mechanism via crowdsourcing for smart surveillance. Comput. Commun. **152**, 149–154 (2020)
9. Lin, Q., Deng, L., Sun, J., Chen, M.: Optimal demand-aware ride-sharing routing. In: IEEE Conference on Computer Communications, pp. 2699–2707 (2018)
10. Liu, C.H., et al.: Curiosity-driven energy-efficient worker scheduling in vehicular crowdsourcing: a deep reinforcement learning approach. In: IEEE International Conference on Data Engineering, pp. 25–36 (2020)
11. Liu, Q., Peng, J., Ihler, A.T.: Variational inference for crowdsourcing. In: Advances in Neural Information Processing Systems 25 (2012)
12. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
13. Ni, W., Cheng, P., Chen, L., Lin, X.: Task allocation in dependency-aware spatial crowdsourcing. In: IEEE International Conference on Data Engineering, pp. 985–996 (2020)
14. Rana, R.K., Chou, C.T., Kanhere, S.S., Bulusu, N., Hu, W.: Ear-phone: an end-to-end participatory urban noise mapping system. In: ACM/IEEE International Conference on Information Processing in Sensor Networks, pp. 105–116 (2010)
15. Shen, W., He, X., Zhang, C., Ni, Q., Dou, W., Wang, Y.: Auxiliary-task based deep reinforcement learning for participant selection problem in mobile crowdsourcing. In: ACM International Conference on Information & Knowledge Management, pp. 1355–1364 (2020)
16. Wang, H.N., et al.: Deep reinforcement learning: a survey. Front. Inf. Technol. Electr. Eng. **21**(12), 1726–1744 (2020)
17. Xiong, F., Xu, S., Zheng, D.: An investigation of the uber driver reward system in china-an application of a dynamic pricing model. Technol. Anal. Strat. Manage. **33**(1), 44–57 (2021)

18. Xu, L., Zhou, X.: A crowd-powered task generation method for study of struggling search. Data Sci. Eng. **6**(4), 472–484 (2021)
19. Ye, G., Zhao, Y., Chen, X., Zheng, K.: Task allocation with geographic partition in spatial crowdsourcing. In: ACM International Conference on Information & Knowledge Management, pp. 2404–2413 (2021)
20. Yin, B., Li, J., Wei, X.: Rational task assignment and path planning based on location and task characteristics in mobile crowdsensing. IEEE Trans. Comput. Soc. Syst. **9**(3), 781–793 (2022)
21. Zhao, Y., Zheng, K., Cui, Y., Su, H., Zhu, F., Zhou, X.: Predictive task assignment in spatial crowdsourcing: a data-driven approach. In: IEEE International Conference on Data Engineering, pp. 13–24 (2020)
22. Zheng, L., Cheng, P., Chen, L.: Auction-based order dispatch and pricing in ridesharing. In: IEEE International Conference on Data Engineering, pp. 1034–1045 (2019)