World Wide Web (2025) 28:13
https://doi.org/10.1007/s11280-024-01325-9

A Dual-Embedding Based Reinforcement Learning Scheme 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] · Guihai Chen[2]

Received: 14 February 2023 / Revised: 28 November 2024 / Accepted: 17 December 2024
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

Abstract
With the popularity of mobile devices, spatial crowdsourcing has attracted widespread attention, which collects spatial tasks with location constraints and assigns them to workers who can travel to certain locations to participate in and obtain profits. One of the core issues is task assignment, in which tasks should be assigned to proper workers to maximize the overall utilities. In the paper, we consider a Utility-driven Destination-aware Spatial Task Assignment (UDSTA) problem, where the utility of a worker is modeled as the completed task profit minus the worker's travel cost, which is more realistic and involves route planning while assigning tasks. We prove that this problem is NP-complete and propose a dual-embedding based deep Q-Network (DE-DQN) to sequentially assign tasks to proper workers. Specifically, we design a utility embedding to reflect the top-k utility tasks for workers and worker-task pairs, and a coverage embedding to represent the potential future utility of an assignment action. The state of DQN consists of the utility embedding, remaining workload, and cumulative utility. Besides, the action of this DQN is formed by concatenating the utility and coverage embedding. We also provide an enhanced version called DE-Rainbow by using Rainbow DQN instead of traditional DQN for further optimization. For the first time, we combine the dual embedding with DQN to achieve a multi-task and multi-worker matching and obtain the route plans of workers. Experiments based on both synthetic and real-world datasets indicate that DE-DQN and DE-Rainbow perform well and show significant advantages over the baseline methods.

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

1 Introduction

Spatial crowdsourcing (SC) has emerged as a new working mode in recent years. It breaks down complex tasks from task publishers into multiple small and simple tasks that carry location information. These tasks are then allocated to a large number of mobile workers with

This article belongs to the Topical Collection: *APWeb-WAIM 2022*
Guest Editors: Calvanese Diego, Toshiyuki Amagasa and Bohan Li

Extended author information available on the last page of the article

diverse backgrounds, who can efficiently and quickly reach the specific locations to finish the tasks [1]. With the prevalence of smart mobile devices and the progress of wireless network technology, spatial crowdsourcing is permeating many aspects of society and profoundly altering human lifestyles. Numerous related applications based on spatial crowdsourcing systems have been proposed and put into practice, such as Common Sense [2] for pollution data collection, Ear-phone [3] for urban noise sensing, GBus [4] for real time bus information obtaining, Jam Eyes [5] for traffic jams detection, and Uber [6] for taxis hailing.

One of the core issues in spatial crowdsourcing is *task assignment*. The purpose is to allocate tasks with unique attributes such as revenue and location to the most appropriate workers with characteristics such as service scope and ability [7]. Such a worker-task matching process can be achieved via a spatial crowdsourcing platform, as shown in Figure 1. Requesters submit tasks to be completed with various requirements to this spatial crowdsourcing platform. Simultaneously, workers join in the platform through recruitment. The platform then matches compatible tasks with workers according to the information from task pool and worker pool. If the quantities of tasks and workers are small, the assignment problem would be easy to deal with even with an enumeration or brute force method. Nevertheless, if the scales of tasks and workers increase, the computational complexity of the problem may increase polynomially. This is because a valid assignment needs to take all possible combinations of tasks and workers and corresponding route planning solutions into account. Therefore, proposing efficient and effective methods for large-scale task assignment problem in SC deserves in-depth research.

In this paper, we focus on the task assignment issue. Here, multiple tasks and workers are distributed in a spatial area. A worker has a limited capacity, meaning that the worker can only receive and complete a certain number of tasks, while a task can only be assigned to one worker. When a worker travels to a task location, a travel cost occurs, which is often assumed to be proportional to the distance between them [8]. When a task is completed, the worker will receive a corresponding profit. Since a worker always wants to maximize his/her profit while minimize the travel cost, we can define a worker's *utility* as the overall profit of his/her completed tasks minus the overall travel cost [9]. Then, given the initial positions of all workers and tasks, and the corresponding task completion profits, the goal of our problem is to match each required task to an appropriate worker with a valid route plan, so as to maximize the total utility of all workers. Such problem is defined as Utility-driven

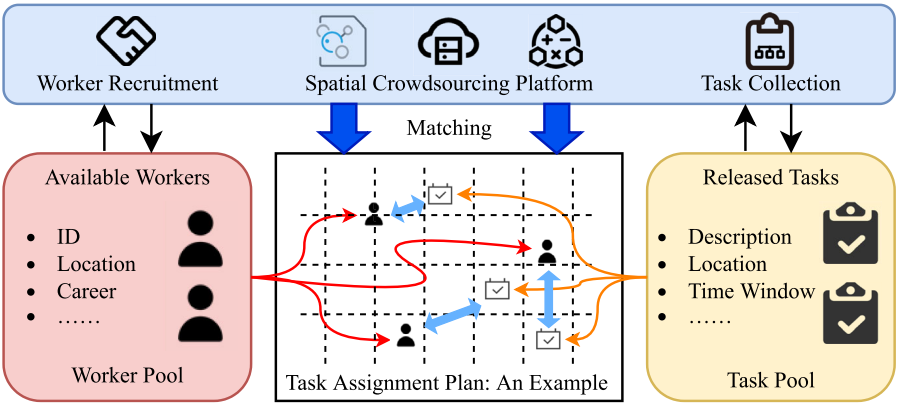


Figure 1 An illustration of task assignment process in spatial crowdsourcing

Destination-aware Spatial Task Assignment problem (UDSTA). For the problem, we consider the situation where the number of tasks is less than the number of workers multiplied by the maximum worker's capacity, whose rationality is discussed in detail in Section 3.2.

Figure 2 exhibits a toy example showing the initial positions of 2 workers w_1, w_2 and 9 tasks $s_1, s_2, \cdots, s_9$, where the radius of nodes represents the profit of tasks, and the length of edges represents the distance between two locations. Assuming that the two workers' cost are equal to the distance of the path, the proper task sequence for worker w_1 with capacity $c_1 = 3$ will be a route plan written by $r_1 = [s_2, s_3, s_5]$. In fact, worker w_1 chooses a 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 exists better neighboring tasks around s_8 compared with s_6. However, computing such neighborhood structures deterministically is somewhat similar to a flooding algorithm, with impractical exponential computation complexity.

Among the previous work, [9] proposes a Utility Priority algorithm by greedy search for high utility tasks. However, the method fails to address the difficulty shown in Figure 2. [10] considers the task similarity and proposes efficient heuristic algorithms for the single-worker and multi-worker scenarios, yet it does not consider the potential future utilities of the current assignment. These traditional algorithms may fall into local optimum and cannot obtain global optimum solutions [11]. Even worse, in the large-scale scenario, solving the problem becomes extremely time-consuming [12]. Hence, to leverage the advantages of machine learning techniques in finding global optimal solutions and handling large-scale problems, we propose a new value-based deep reinforcement learning framework called dual-embedding based deep Q-Network (DE-DQN). DE-DQN innovatively employs dual embedding vectors as the input of DQN to learn the optimal strategy of task assignment. Specifically, We design the utility embedding to reflect the information of top-k utility tasks for workers and worker-task pairs, which measures the possible profit of workers and worker-task pairs in the current state. Next, the coverage embedding is designed to represent the information of potential future utility of workers with an assignment action. Accordingly, we concatenate the utility embedding, the remaining workload capacity, and the cumulative utility to form the state of DQN, which represents the task environment around the workers. Besides, we concatenate the utility embedding of worker, task, potential future task, and the coverage embedding to represent the action of DQN, assigning a task to a worker at each step. The goal of the DQN framework is to identify the best action from the action space. An enhanced method called DE-Rainbow is proposed using the Rainbow DQN to replace the

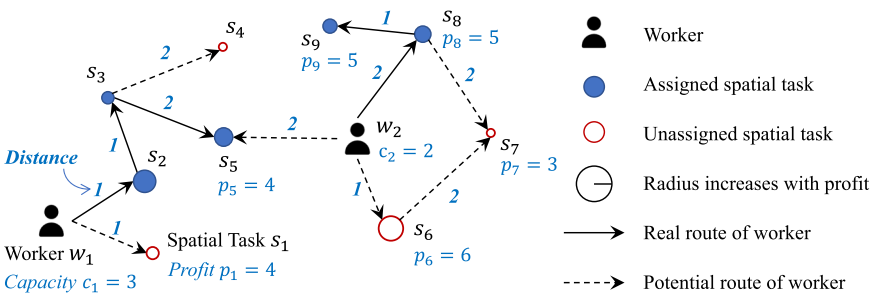


Figure 2 A toy example with 2 workers and 9 spatial tasks

original DQN. The Rainbow DQN provides important extensions including DDQN, Dueling DQN, prioritized replay, distributional RL and Noisy Net [13].

The experimental results based on both synthetic and real-world datasets indicate that DE-DQN and DE-Rainbow can significantly improve the efficiency of problem solving and performs well in large-scale scenarios. What's more, since UDSTA problem is a generic setting for task assignment problem, the proposed DE-DQN and DE-Rainbow can be easily adapted to address many variant problems such as sweep coverage problem [14] and vehicle dispatching problem [15] by adding time-related factors and changing the goal.

To summarize, our contributions are presented as follows:

- We precisely formulate the task assignment problem with utility and destination properties in spatial crowdsourcing (named UDSTA problem), and prove its NP-completeness.
- We design a dual embedding integrated Deep Q-Network, and propose the DE-DQN and DE-Rainbow method to solve the UDSTA problem.
- We compare DE-DQN and DE-Rainbow with other baseline methods through experiments on synthetic and real-world datasets and present the advantages of our methods.

The rest of the paper is organized as follows. Section 2 introduces the related work of task assignment problem and DRL for crowdsourcing. Section 3 provides preliminaries and problem statement. Section 4 introduces the design of dual embedding. Section 5 and Section 6 explains the framework and training of DE-DQN and DE-Rainbow respectively. Section 7 presents the experiment results compared to baseline methods. Section 8 makes conclusion.

2 Related work

2.1 Task assignment problem

Task assignment problem in crowdsourcing has been extensively paid attention to and researched in the past ten years as most worker-task matching requirements in real world application can be formulated as this problem [26]. There are different kinds of optimization objectives including task number [24], quality of service [16], fairness [25], platform revenue [18], and matching stability [27]. For example, [28] aims at minimizing the maximum travel time and total latency of the requests. [29] aims at minimizing the sum of the travel time and penalty. In our discussion, we aim at maximizing the overall utility.

Several studies have also discussed different constraints on the task assignment problem. [9] defines a team-oriented scenario, considering the worker's skill constraint of tasks. [30] considers the pick-up and drop-off problem in the car-hailing scenario. In our discussion, we consider a scenario where workers can complete tasks directly at the task location.

Multiple methods based on different technologies have been studied and mentioned in recent works [31]. A common model employed for solving task assignment problem is bipartite graph [32], in which tasks and workers are abstracted into nodes of both sides. [22] 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. Besides traditional methods, [23] first presents a deep reinforcement learning framework on vehicular crowdsourcing problem, both maximizing task accomplishment and coverage fairness, and minimizing the energy cost.

To be detailed, Table 1 provides an overview of the work of different literatures solving task assignment problem including scenario, goal, mode, side, model and method.

Table 1 Related work for task assignment in spatial crowdsourcing

Literature	Scenario	Goal	Mode	Side	Model	Method
[16]	Worker recruitment	Max completion quality	Online	Requester	Combinatorial MAB	Upper confidence bound
[17]	Predictive TA	Max task assignment	Online	Worker	Data driven	Greedy/Graph partition
[18]	Ride sharing	Min cost	Online	Both	Stochastic OPT	Demand-aware
[9]	Skill aware	Max task completion	Offline	Worker	/	Greedy heuristic
[19]	Quality aware	Max task completion	Online	Both	Evaluation metrics	Linear algorithm
[11]	Dependence aware	Max task assignment	Offline	Worker	Approximation	Greedy/Game theory
[20]	Cooperative aware	Max cooperation quality	Offline	Worker	/	Greedy/Game theory
[21]	Quality aware	Max gain	Online	Worker	Latent topic	Greedy
[22]	Task selection	Max completion quality	Offline	Worker	Task-time route	Bayesian potential game
[23]	Vehicular crowdsourcing	Max task completion	Offline	Worker	Curiosity driven	Deep RL
[24]	Preference aware	Max task assignment	Online	Worker	Min cost max flow	Greedy
[25]	Location aware	Max task assignment	Online	Both	Lyapunov OPT	Greedy

2.2 DRL for crowdsourcing

Reinforcement learning (RL) has become a hotspot in research field recently. Some variants of RL also emerge 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 [33], 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 [34], which utilizes convolutional neural network and Q-learning [35], and performs well in many applications including task arrangement in crowdsourcing [36]. In addition to value-based DRL, policy-based DRL presents a significant advantage in continuous action space problem, such as deep deterministic policy gradient algorithm (DDPG) [37].

In recent years, some works have used deep reinforcement learning to solve the task assignment problem in the spatial crowdsourcing [38]. [39] presents a spatial crowdsourcing problem which allows but not forces the workers to cooperate. A Multi-agent DRL Model on the base of Advantage Actor-Critic (A2C) is proposed in this work, involving attention mechanism to utilize the information of other workers. [40] studies spatial crowdsourcing in drone data collection and promotes Fully Decentralized Multi-Agent Proximal Policy Optimization (FD-MAPPO). [41] pays attention to environment sensing crowdsourcing with a method of combining DRL with 3-dimensional Convolutional Neural Network (CNN). [42] proposed a Deep Deterministic Policy Gradient (DDPG) based framework to gain a high revenue in each round. However, they choose an assigning action without considering the effect on future revenue. Shan et al. proposed an attention mechanism to the future revenue [43]. [44] introduces geographic partition into spatial crowdsourcing and conducts the RL method to solve the problem. [45] proposes auxiliary-task based DRL to achieve multi-objective optimization in participant selection problem of mobile crowdsourcing.

In these previous works, they do not use explicit designs, such as eigenvectors, to learn the effect of assigning actions on future revenue. Here, we propose a dual embedding method, which explicitly considers the future impact of an assigning action when designing the state and action of DQN, reducing the difficulty of DQN learning. Table 2 illustrates the classification of reinforcement learning models and the corresponding typical models, while Table 3 shows the comparisons of goal, method, future revenue and explicit expression. As assigning tasks to appropriate workers corresponds to a discrete action space, we can calculate the reward for each assignment without having to build a model to reflect the response of the environment to actions, so we choose DQN to solve the UDSTA problem. What's more, most

Table 2 Classification of RL models and corresponding typical models

Criteria 1	Criteria 2	Typical Models
Model-Free	Policy Optimization	Policy Gradient^C, A3C [46], TRPO [47], PPO [48]
	Q-Learning	Q-Learning^D, DQN^D [49]
	Policy Opt+Q-Learning	DDPG^C [50], TD3^C [51], SAC^C [52]
Model-Based	Learn the Model	I2A [53], MBMF [54]
	Model is Given	AlphaZero [55]

*A^C means that the model A is suitable for the continuous action and A^D means that the model A is suitable for the discrete action.

Table 3 Comparisons on literature of DRL for crowdsourcing

Literature	Goal	Method	Highest gains (Breadth)	Long-term gains (Depth)
[39]	Max total utility	Multi-agent DRL	✓	
[40]	Max total utility	FD-MAPPO	✓	
[41]	Max data collection	DRL+CNN	✓	
[42]	Max total utility	DDPG	✓	
[44]	Max task allocation	Attention+GNN		✓
[23]	Max task completion	DRL	✓	
[43]	Max total utility	Attention		✓
[45]	Max worker utility	ADRL,	✓	
Ours	Max worker utility	Rainbow DQN	✓	✓

previous works take the future revenue into concern but explicit expression in our paper is not involved.

3 Preliminaries and problem statement

In the paper, we mainly focus on the scenarios such as region monitoring, tourist photoing, household service, etc., where the location distribution of tasks and workers has more impact on the solutions, and time-related constraints are less sensitive (e.g., the time windows can be counted by days, or weeks, which are sufficient for the computation of a task assignment algorithm). In such scenarios, tasks are scattered in different locations. To obtain profits, workers are motivated to complete a set of tasks (whose size is denoted as the capacity of the worker). Meanwhile, workers need to bear the costs such as traveling expense when completing tasks. Hence, the utility of a worker is equal to the profit minus the cost and the objective of a task assignment problem is to maximize the overall utility for all workers. For time sensitive scenarios, time-related factors should be considered to the assignment and thus included in the DQN framework, which we plan to investigate in future work.

3.1 Extensions to DQN

In addition to standard DQN, many other improved methods and extensions were raised. Since DQN often selects over-valued actions, Double DQN is proposed in [56] to select appropriately valued actions by constructing a Q' function. In addition, Dueling DQN uses a state function and an action advantage function to learn the value of the static environment itself and the additional value brought by the action respectively to achieve better stability than DQN that directly evaluates the Q-value of the action [57]. Besides, Compared with the random sampling used by DQN, Prioritized Experience Replay is proposed in [58] to samples more data with larger TD error, so that the model can learn better and faster. Noisy net is introduced in [59] to increase the exploration ability of DQN. Compared with ϵ-greedy that only explores more actions, noisy net has a stronger exploration ability by adding noise to the parameters for state-independent consistent exploration. Distributed RL makes the reward obtained by the model more realistic by estimating the distribution of the reward instead of the

expected reward [60]. Therefore, we propose an enhanced framework by using the Rainbow DQN which combines the above extensions [13]. Table 4 summarizes the extensions and the corresponding advantages to DQN.

3.2 Problem formulation

Following most existing works in crowdsourcing [61], we adopt the round-based model to formulate the UDSTA problem. In each round, the platform operates on a set of pended tasks and workers. Some basic definitions related to worker, task, and utility are introduced as follows.

Definition 1 (Spatial task) A spatial task s_j is denoted by a tuple $\langle l_j^s, p_j \rangle$, where l_j^s is the geographical location of task s_j; p_j is the profit when task s_j is completed. Note that, in the spatial crowdsourcing, a task can only be completed when the worker is at the location of the task.

Definition 2 (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 is an ordered task sequence consisting of tasks assigned to w_i; C_i is w_i's capacity limit of workload.

Definition 3 (Worker's utility for completing task) The utility of worker w_i to complete 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 worker w_i to complete task s_j.

Definition 4 (Overall utility) The overall utility of a round 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 tasks and $cost(w_i) = \sum_{s_j \in r_i} cost(w_i, s_j)$ represents the overall cost of worker w_i.

Table 4 Extensions to DQN

Extensions	DQN Drawbacks/ Components	Descriptions of extensions	Advantages
Double DQN [56]	select over-valued actions	design Q function	select proper-valued actions
Dueling DQN [57]	action evaluation function	design state & action function	achieve better stability
Prioritized Replay [58]	random sampling	sample data with large TDE	learn better/faster
Noisy net [60]	ϵ-greedy: explore more actions	add noise to parameters	stronger exploration ability
Distributed RL [59]	obtain expected reward	obtain reward distribution	More in line with reality

By default, U_{all} should be positive after assignment, otherwise workers would work at a sacrifice, which is impractical apparently. We also 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, which means the destination of s_j should be included in r_i if it is assigned to w_i.
- **Capacity constraint:** Number of tasks assigned to w_i cannot exceed the workload capacity limit C_i. C_i should be a positive integer, usually setting as a number within [3, 10] in practice.

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 the work, we assume the $cost(w_i, s_j)$ to be proportional to the distance $dist(l_i^w, l_j^s)$ between them referring to [8], 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. Here, we consider the Euclidean distance between pairwise points.

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.

In some literature, task assignment problem is also referred to as a relaxed *matching* problem or *covering* problem, because we are pairing tasks with workers, but usually achieving a multiple-to-one mapping according to the worker's capacity constraint. Accordingly, some design of our embedding methods are motivated from the traditional combinatorial algorithms solving matching or covering problems, which will be further discussion in Section 4.

Another thing worth discussing is the number of workers, or the cardinality $|\mathcal{W}|$ of worker set $\mathcal{W}$, versus the number of tasks (the cardinality $|\mathcal{S}|$ of task set $\mathcal{S}$). Although intuitively we think $|\mathcal{W}| > |\mathcal{S}|$, since a crowdsourcing platform usually has thousands of registered workers, practically the number of active workers are far less than that of registered ones, so in our discussion, each round we may have

$$|\mathcal{S}| > |\mathcal{W}| \cdot \max_{w_i \in \mathcal{W}} \{C_i\}. \tag{4}$$

Inequality (4) implies that we do not need to consider the completeness of tasks or the maximization of finished tasks s_j^*, which may bring a multi-objective optimization problem, since once we maximize the overall utility in each round, we have successfully "assigned" all possible workers, inducing a maximum number of finished tasks simultaneously.

3.3 NP-completeness analysis

Also the definition of UDSTA problem is easy to understand, it is indeed an NP-Complete problem. We prove its NP-completeness through a polynomial time many-to-one 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 in [62].

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 polynomial time many-to-one reduction from TSP-Path, written as $TSP\text{-}Path \leq_m^p UDSTA$.

Given any instance of TSP-Path problem, where there are n cities forming a graph $G = (V, E)$ with $|V| = n$. We have a distance function $d(e) \in \mathbb{Z}^+$ for each $e = (v_i, v_j) \in E$, and a positive integer K. The decision version of this problem wants to answer "Is there a path with n vertices having length K or less, i.e., a permutation $[v_{\pi(1)}, v_{\pi(2)}, \cdots, v_{\pi(n)}]$ of V such that $\sum_{i=1}^{n-1} d(v_{\pi(i)}, v_{\pi(i+1)}) \leq K$." Now we can construct a special instance of UDSTA problem. Let $\mathcal{S} = V$ (each with its corresponding location l_j^s). The profit p_j of each task is sufficiently large, i.e., a positive P uniquely. Let $\mathcal{W} = \{w_1, w_2\}$ with two workers, whose positions are the same as v_1, say, $l_1^w = l_2^w = l_1^s$. Define the cost functions of each pairwise w_i and t_j as the distance function $d(\cdot, \cdot)$ with the same locations, and the capacities of each worker are $C_1 = C_2 = |V|$. The cost per unit travel distance β_1 and β_2 are set as 1. Given another positive integer $K' = n \cdot P - K$, the decision version of UDSTA problem asks whether we have a task assignment strategy with total utility larger than of equal to K'.

Since the workers want to earn more profit while pay less cost, the maximum profit in this case should be $n \cdot P$, meaning all tasks are assigned successfully. Then based on (2), maximizing utility is converted into minimizing the overall route length. Now we are able to conclude in polynomial time that: if the TSP-Path instance has a "yes" solution with K, then the constructed UDSTA instance has a "yes" solution with $K' = n \cdot P - K$. It is easy to plan two routes for w_1 and w_2 along the TSP path starting from v_1, and traveling to two opposite directions. If v_1 is the starting vertex of this TSP path, then only w_1 needs to follow the TSP path, and w_2 can stay at where it is. Reversely, if this special UDSTA instance has a "yes" solution, the route plans of two workers must form a path including each vertex one and only once. Otherwise any other paths would bring an overall cost greater than K. Thus, the combination of two route plans together forms a TSP path successfully, which gives a "yes" answer to the original instance of TSP-path. According to Karp's definition of the NP-Completeness reduction, we have finished this proof. □

The important notations in the rest of the paper are summarized in Table 5.

4 Dual-embedding

Since the travel cost is assumed to be proportional to distance between worker and task, we utilize the graph to model the destination-aware spatial crowdsourcing. To solve the UDSTA problem, we propose a neural network-based learning method called DE-DQN, and an enhanced version called DE-Rainbow. We first describe the dual-embedding method to obtain the vector representations of workers and tasks for flexible computation in the neural network-based learning in Section 4. Then, we use a Deep Q-Network (DQN) to

Table 5 Definitions and notations

Symbol	Definition
$w_i, \mathcal{W}$	a worker $w_i = \langle l_i^w, r_i, C_i \rangle$; all workers form a worker set $\mathcal{W}$, $w_i \in \mathcal{W}$
$s_j, \mathcal{S}$	a spatial task $s_j = \langle l_j^s, p_j \rangle$; all tasks form a spatial task set $\mathcal{S}$, $s_j \in \mathcal{S}$
$s_j^*, \mathcal{S}^*$	an assigned spatial task s_j^*; all assigned tasks form a set $\mathcal{S}^*$, $s_j^* \in \mathcal{S}^*$
l_i^w, l_j^s	location of worker w_i and task s_j
r_i	task sequence received by worker w_i
C_i	capacity limitation of task assigned to worker w_i
p_j	profit when task s_j is completed
$dist(\cdot, \cdot)$	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, $U_{all} = \sum_{s_j^*} p_j - \sum_{s_j^*} cost(w_i, r_i, s_j)$

sequentially assign tasks to proper workers and obtain the global task assignment solution in order to achieve the objective of maximizing the overall utility in Section 5. We also give an enhanced version called DE-Rainbow by using the Rainbow DQN to replace the DQN Section 6.

We think that numerous workers and tasks constitute a crowdsourcing graph. The graph consists of a large number of nodes with worker and task attributes and edges with distance information, which makes the graph complex and difficult to directly perform DRL to assign tasks. Therefore, we hope to first extract the important information for task assignment from the graph and design the components of DQN based on the information.

As is shown in Figure 2, A temporary low utility assignment action may achieve a higher utility global assignment because of its proximity to another high utility assignment action. We define the impact of an assignment action on global utilities as potential future utility, which should be reasonably considered to obtain higher global utilities.

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. For a worker, we can select the top-k neighboring tasks by sorting the utility. Therefore, we use the utility embedding to reflect the top-k neighboring tasks' utilities from the breadth perspective and the coverage embedding to represent the potential future utility of an assignment linked with the remaining capacity of the worker from the depth perspective, which makes the learning of DQN feasible.

4.1 Utility embedding

In our scenario, the state of the DQN represents the environment containing information about optional workers and uncompleted tasks, and the action represents matching a worker with an uncompleted task, so we design two types of utility embedding for state and action. Here, we use the worker-task pair (w_i, s_j) to represent an assignment decision. For the state, we design the utility embedding to reflect the top-k neighboring utility tasks for worker w_i according to the worker's historical record, current location, and neighboring uncompleted

tasks. For the action, we similarly design the utility embedding for assignment decision (w_i, s_j) to reflect the top-k neighboring utility tasks when s_j is assigned to w_i. Compared with the greedy-based and heuristic algorithms, the utility embedding could be helpful to provide a larger feasible search space because it considers the top-k assignment decisions with potential in the future, rather than only the temporarily optimal decision.

4.1.1 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

$$\mathbf{V}_{w_i}^u = \left(u(w_i, s_{w_i^1}), u(w_i, s_{w_i^2}), \cdots, u(w_i, s_{w_i^k})\right) \tag{5}$$

4.1.2 Representation vector for worker-task pair

An assignment can be seen as a worker-task pair (w_i, 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_i, s_j) is

$$\mathbf{V}_{w_i s_j}^u = \left(u_{w_i}(s_j, s_{(w_i s_j)^1}), u_{w_i}(s_j, s_{(w_i s_j)^2}), \cdots, u_{w_i}(s_j, s_{(w_i s_j)^k})\right) \tag{6}$$

4.2 Coverage embedding

The coverage embedding is designed to represent the long-term potential future utility of an assignment decision from the depth perspective, which is helpful to estimate the value of an action of DQN. Specifically, for the assignment decision (w_i, s_j), we define the propagation chain with the remaining workload capacity rc_i as length to represent the possible future tasks assigned to w_i. Therefore, the long-term potential future utility could be denoted by the sum of the highest neighboring task utilities in the propagation chain. We define $s_{(w_i s_j)^1}^m$ to represent the highest utility task corresponding to the m_{th} position of the propagation chain. For instance, $s_{(w_j s_j)^1}^2 = s_{(w_i s_{(w_i s_j)^1})^1}$. Especially, $s_{(w_i s_j)^1}^0 = s_j$. Hence, for pair (w_i, s_j), we can express the long-term potential future utility of assigning $s_{w_i s_j}^k$ to w_i as

$$u_{w_i}^c(s_j, s_{(w_i s_j)^k}) = u_{w_i}(s_j, s_{(w_i s_j)^k}) + \sum_{m=1}^{rc_i - 1} u_{w_i}(s_{(w_i s_j)^1}^{m-1}, s_{(w_i s_j)^1}^m) \tag{7}$$

Therefore, for the assignment decision (w_i, s_j), we concatenate the top-k highest task utilities to form the coverage embedding $\mathbf{V}^c(w_i, s_j)$ as

$$\mathbf{V}^c(w_i, s_j) = \left(u_{w_i}^c(s_j, s_{(w_i s_j)^1}), \cdots, u_{w_i}^c(s_j, s_{(w_i s_j)^k})\right) \tag{8}$$

5 Framework and training for DE-DQN

Based on the discussion in Section 3.1, DQN is adopted to solve the UDSTA problem. We propose a dual-embedding based DQN method called DE-DQN using the combination of the above obtained dual embedding and DQN. Figure 3 shows the framework diagram of DE-DQN (without extensions).

5.1 DQN components

The components of DE-DQN are described as follows.

5.1.1 State

We design the state to reflect the existing worker-task matching and the availability of workers to take on tasks in the spatial crowdsourcing, which could be measured from three perspectives: potential future utility, remaining workload capacity and cumulative utility. Specifically, the state vector concatenates the above three parts. The first part is the mean of the top-k utilities for all workers $\mathbf{V}_W^u = \frac{1}{|W|}\sum_{w_i \in W} \mathbf{V}_{w_i}^u$. 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 $uc = (\sum_{s_j \in \mathcal{S}^*} p_j, \sum_{w_i} cost(w_i))$. Hence, the state vector $x_s = (\mathbf{V}_W^u \| rc \| uc)$.

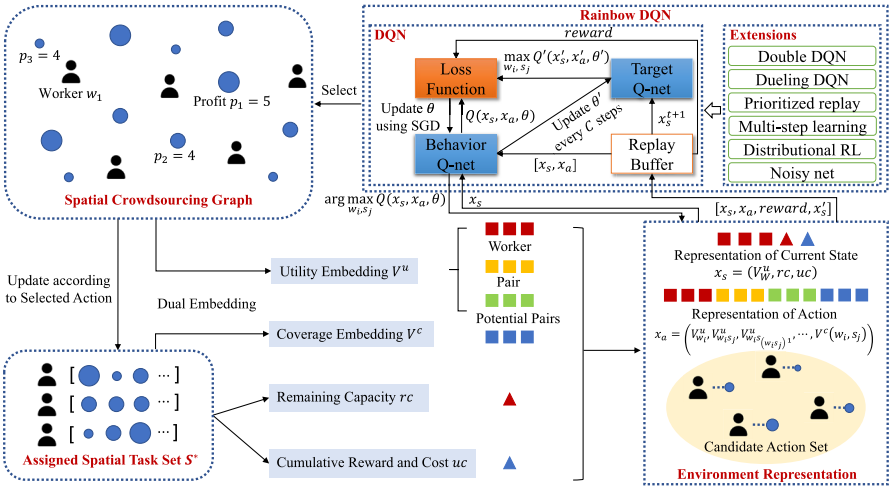


Figure 3 The illustration of DE-DQN and DE-Rainbow. Without extensions, DE-DQN first calculates the utility embedding $\mathbf{V}^u$ and coverage embedding $\mathbf{V}^c$ for workers and pairs. The state representation x_s is the concatenation of three parts: mean of the utility embedding for workers $\mathbf{V}_W^u$, 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. DE-Rainbow uses the Rainbow DQN including extensions such as Double DQN and Noisy net to replace the DQN

5.1.2 Action

The action represents an assignment decision (w_i, s_j). We measure the value of an action in terms of both depth and breadth, which means we combine the utility embedding (6) and coverage embedding (8) of an assignment decision to design the action of DE-DQN. Besides, we consider the potential future utility of the corresponding possible next assignment. Specifically, we concatenate the utility embedding of the assignment (w_i, s_j), potential future assignments $\{(w_i, s_{(w_i s_j)^1}), \cdots, (w_i, s_{(w_i s_j)^k})\}$, and coverage embedding of (w_i, s_j) to indicate the action $x_a = (\mathbf{V}^u_{w_i} \| \mathbf{V}^u_{w_i s_j} \| \mathbf{V}^u_{w_i s_{(w_i s_j)^1}} \| \cdots \| \mathbf{V}^c(w_i, s_j))$.

5.1.3 Reward

The reward represents the utility obtained by an assignment action. When assigning task s_j to worker w_i, the reward is equal to $p_j - cost(w_i, s_j)$.

5.1.4 Policy

Since DQN is an off-policy algorithm, a neural network should be employed to approximate the Q-value of an action. In our DE-DQN, we take different algorithms in the training phase and the testing phase. Specifically, the ϵ-greedy algorithm is adopted in the training stage to do more exploration, while the pure greedy algorithm is adopted in the testing stage to maximize the overall utility.

The ϵ-greedy policy in the training stage is

$$\gamma_{\epsilon-greedy}(x_s) = \begin{cases} \underset{((w_i,s_j)|s_j \in \mathcal{S}-\mathcal{S}^*)}{\arg\max} Q & \text{w.p. } \epsilon \\ \text{a random feasible pair } ((w_i, s_j)|s_j \in \mathcal{S} - \mathcal{S}^*) & \text{o.w.} \end{cases} \tag{9}$$

The pure greedy policy in the testing stage is

$$\gamma_{greedy}(x_s) = \underset{((w_i,s_j)|s_j \in \mathcal{S}-\mathcal{S}^*)}{\arg\max} Q \tag{10}$$

5.2 Behavior and target Q-network

The DE-DQN framework consists of a behavior Q-network and a target Q-network with the same structure and different parameters 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. The difference is that the behavior Q-network is used to evaluate the value of the action in the current state, while the target Q-network is used to evaluate the value of the action in the next state. Besides, the parameters of the target Q-network remains unchanged in C training steps to reduce the correlation between the predicted Q-value and the target Q-value. After C steps, the parameters of the behavior Q-network are copied to update the parameters of the target Q-network.

Specifically, 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 could 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 $L(\theta)$:

$$L(\theta) = \mathbf{E}[(\hat{y^t} - Q(x_s^t, x_a^t, \theta))] \tag{11}$$

$$\hat{y^t} = reward^t + \eta \max_{(w_i, s_j)} Q'(x_s^{t+1}, x_a^{t+1}, \theta') \tag{12}$$

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.

5.3 Training process

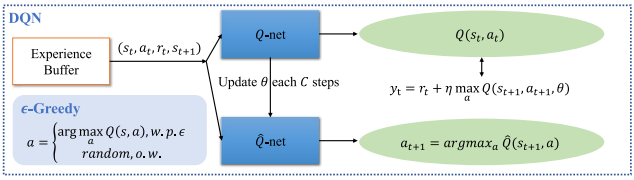
Algorithm 1 DE-DQN Agent Training.

Require: Spatial crowdsourcing graph, worker set $\mathcal{W}$, task set $\mathcal{S}$
Ensure: Parameters of Q-network
1: Initialize replay buffer D;
2: Initialize parameters of the behavior and the target Q-networks, $\theta = \theta'$;
3: **for** episode $\leftarrow$ 1 to E **do**
4: Initialize the assigned spatial task set $\mathcal{S}^* = \emptyset$;
5: **for** $t \leftarrow 1$ to $\sum_{w_i} C_i$ **do**
6: Select action by $\gamma_{\epsilon-greedy}$ policy;
7: Calculate $reward^t$;
8: $x_s^{t+1} \leftarrow f_t(x_s^t, x_a^t)$;
9: Store $[x_s^t, x_a^t, reward^t, x_s^{t+1}]$ into D;
10: Randomly sample a minibatch data from D;
11: **if** $t = k$ **then**
12: $y^t = reward^t$;
13: **else**
14: $y^t = reward^t + \eta \max_{(w_i, s_j)} Q'(x_s^{t+1}, x_a^{t+1}, \theta')$;
15: **end if**
16: **end for**
17: Use SGD to take a gradient descent step on $(y^t - Q(x_s^t, x_a^t, \theta))^2$;
18: After C steps, update $\theta' = \theta$.
19: **end for**

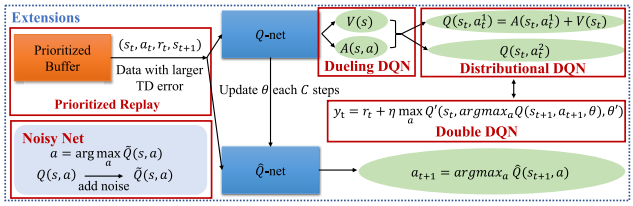
Algorithm 1 shows the training process of DE-DQN. We first initialize the replay buffer D, which is proposed in [34] to reduce the dependency among data, and the parameters of the two Q-networks (Lines $1 - 2$). For each training episode, we initialize the assigned spatial task set (Line 4). The episode ends when the number of assigned tasks reaches $\sum_{w_i} C_i$, the sum of capacity of all workers (Line 5). In each episode, we adopt the $\gamma_{\epsilon-greedy}$ policy to choose an assignment action from the discrete action space (Line 6). After selecting an action, we calculate the next state through a state transition function (Line 8). After obtaining the corresponding reward and the next state, we store the historical record in the replay buffer (Line 9). We update the parameters in Q-network by stochastic gradient descent (SGD) method (Line 15). After C steps, we update the parameters in the target Q-network (Line 16).

6 Framework and training for DE-Rainbow

To better solve the UDSTA problem, we adopt the Rainbow DQN to select the proper worker-task pairs in each step. DE-Rainbow provides some important extensions to the DQN framework, including DDQN, Dueling DQN, prioritized replay, distributional RL, and noisy net [13]. Figure 4a illustrates the structure of the original DQN. Figure 4b demonstrates the



(a) DQN



(b) Extensions

Figure 4 Structural comparisons on various extensions of the DQN

structural variations of the DQN by the above extensions, where the red boxes represent the specific changes for each distinct extension described in Table 4. DDQN is effective in diminishing overestimation bias and enhancing learning stability. Dueling DQN and Prioritized Replay contribute to a boost in learning efficiency. Distributional RL is beneficial for augmenting robustness. Moreover, Noisy Net is conducive to refining exploration strategies and facilitating adaptation to dynamic environments.

We combine the above dual embedding to design the state and action of Rainbow, which is called DE-Rainbow. The state, action and reward representation for DE-Rainbow is the same with the ones of DE-DQN. The differences are that the training policy and structure of deep neural network are changed. Compared to the traditional DQN, we use the noisy net to train the network. Compared to the direct calculation $x_a = Wx_s + b$, the noisy net adds the noisy stream as follows:

$$x_a = (Wx_s + b) + ((W_{noisy} \odot \epsilon^w)x_s + b_{noisy} \odot \epsilon^b) \tag{13}$$

The training objective can be formulated as minimizing the loss function $J(\theta)$ with the multi-step learning:

$$J(\theta) = \mathbf{E}[(\hat{y^t} - Q(x_S^t, x_a^t, \theta))] \tag{14}$$

$$x_a^{t*} = \arg\max_a Q(x_s^{t+1}, x_a, \theta) \tag{15}$$

$$\hat{y^t} = \sum_{k=0}^{N-1} (\gamma^k) reward^t + \gamma^N \max_a Q'(x_S^{t+N}, x_a^{t*}, \theta') \tag{16}$$

$$Q(x_s, x_a, \theta) = V(x_s, x_a, \theta) + A(x_s, x_a, \theta) \tag{17}$$

$$V(x_s, x_a, \theta) = \sum_i z_i \, p_i(x_s, x_a, \theta) \tag{18}$$

$$A(x_s, x_a, \theta) = \sum_i a_i \, p_i(x_s, x_a, \theta) \tag{19}$$

where x_S^t is the state vector at step t, x_a^{t*} is the node representation of the corresponding action, $\gamma \in [0, 1]$ is the discount factor. Equation (15) is to select the node with DDQN. Equation (16) uses the multi-step learning. Q value is calculated base on the dueling network in (17). Equations (18) and (19) are to calculate the Q-value based on the distributional RL. The training process for agent is shown in Algorithm 2.

Algorithm 2 DE-Rainbow Agent Training.

Require: Spatial crowdsourcing graph, worker set $\mathcal{W}$, task set $\mathcal{S}$
Ensure: Parameters of Q-network
Initialize prioritized replay buffer D;
Initialize neural network Q with parameter set θ and target neural network Q' with parameter set $\theta', \theta = \theta'$;
for episode $\leftarrow 1$ to E **do**
  Initialize the assigned spatial task set S as $\emptyset$;
  Initialize the representation vector x_S of the set S;
  for $t \leftarrow 1$ to $\sum_{w_i} C_i$ **do**
    Sample a noisy network ξ;
    Select action by $\arg\max_{x_a} Q(x_s, x_a, \xi, \theta)$;
    Calculate $reward^t$ according to $reward(x_S, x_a) = p_j - cost(w_i, s_j)$;
    $x_s^{t+1} \leftarrow f_t(x_s^t, x_a^t)$;
    Store $[x_s^t, x_a^t, reward^t, x_s^{t+1}]$ into D;
    Randomly sample a minibatch data from D;
    Sample the noisy variable ξ for Q;
    Sample the noisy variable ξ' for Q';
    if $t = k$ **then**
      $y^t = reward^t$;
    else
      Calculate y^t according to (16);
    end if
  end for
  Use SGD to take a gradient descent step on $(y^t - Q(x_s^t, x_a^t, \theta))^2$;
  After C steps, update $\theta' = \theta$.
end for

7 Experiments

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

7.1 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 [63], including a total of 6, 442, 890 check-ins of users from Feb 2009 to Oct 2010. Referred to the practice in [64], we set the distributions of workers and tasks respectively according to the number of check-ins for the Gowalla dataset. Specifically, we choose the locations where the number of user check-ins is greater

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

Table 6 Percentage of tasks in different value ranges

Type	Value				
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	$3.5e^{-2}$	$6.3e^{-3}$	$1.3e^{-3}$
	$4.5e^{-4}$	$2.0e^{-4}$	$3.8e^{-5}$	$2.9e^{-5}$	$1.9e^{-5}$

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 profits are sampled from the real order value distribution in the car-hailing service. Table 6 shows the percentage of tasks in different value ranges in the Didi Chuxing Chengdu dataset, 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. These settings are consistent with the real situation.

As for the SYN dataset, we wish to differentiate from the Gowalla dataset in terms of worker and task distributions in order to test the performance of the proposed model under different distributions. Hence, we generate the locations of 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.

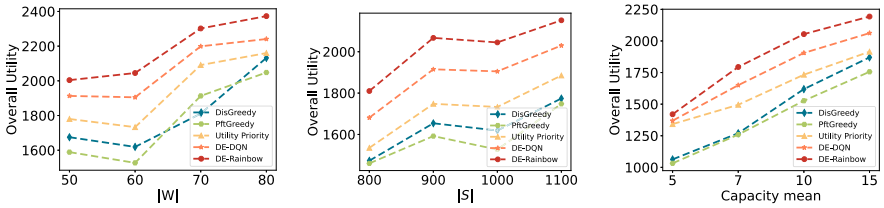
Parameter settings The parameter settings are shown in Table 7, where the default ones are marked in bold. As for training the DE-DQN, we do the training with 500 episodes on randomly sampled data from Gowalla and SYN.

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

- **DisGreedy:** The DisGreedy algorithm is proposed in [10], 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 [10], 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:** [9] proposes the Utility Priority algorithm, matching the largest utility worker-task pair at each step.

Table 7 Parameter setting

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



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

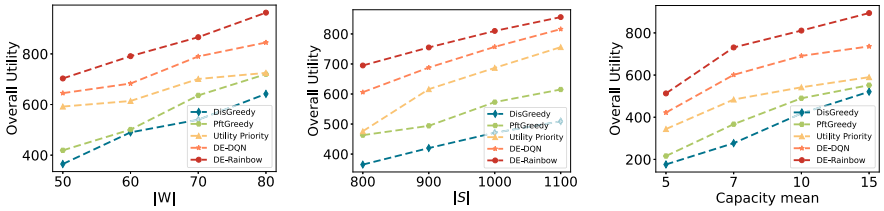
Figure 5 Comparison with various $|\mathcal{W}|$, $|\mathcal{S}|$, and capacity mean on Gowalla

7.2 Performance analysis

All the experiments are implemented on an Apple M1 chip with 8-core CPU and 16GB unified memory. The DE-DQN, DE-Rainbow and baselines are implemented with Python 3.6. Tensorflow 2.0 is used to build the machine learning framework. As shown in Figures 5 and 6, 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-Rainbow, DE-DQN and Utility Priority perform well regardless of the change of cost.

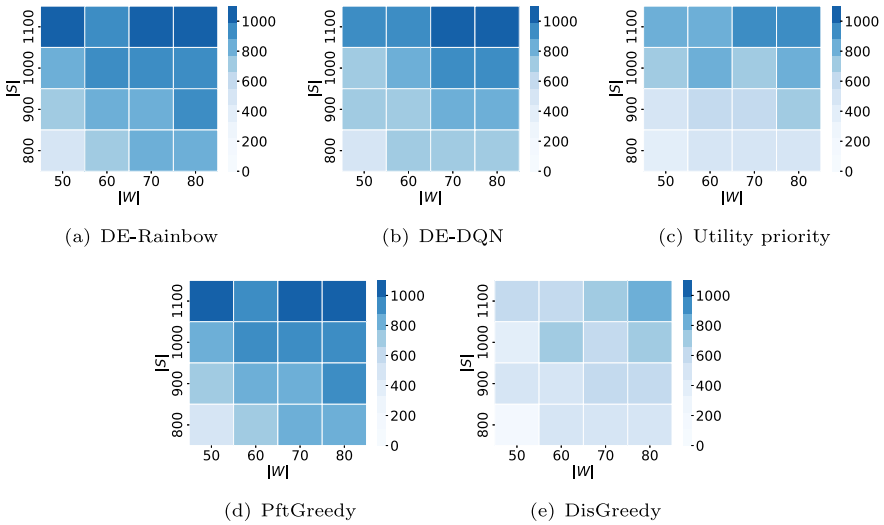
Effect of $|\mathcal{W}|$ As shown in Figures 5a and 6a, the performance of DE-Rainbow and DE-DQN is better than that of other baseline algorithms with various $|\mathcal{W}|$. Simultaneously, the performance gap of DE-Rainbow and DE-DQN over Utility Priority increases as $|\mathcal{W}|$ increases, indicating that DE-Rainbow and DE-DQN are 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 Figures 5b and 6b, DE-Rainbow and DE-DQN achieve the highest overall utility with various $\mathcal{S}$. We note that DE-Rainbow, DE-DQN and Utility Priority can discover better tasks as $|\mathcal{S}|$ increases compared with DisGreedy and PftGreedy. Simultaneously, the performance gap between DE-Rainbow, 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.



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

Figure 6 Comparison with various $|\mathcal{W}|$, $|\mathcal{S}|$, and capacity mean on SYN



(a) DE-Rainbow (b) DE-DQN (c) Utility priority

(d) PftGreedy (e) DisGreedy

Figure 7 Comparison with the overall utility w.r.t. $|\mathcal{W}| \times |\mathcal{S}|$ on SYN

Effect of capacity mean As shown in Figures 5c and 6c, the higher the capacity mean of workers, the better the DE-Rainbow and DE-DQN outperform 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 $|\mathcal{S}|$.

Effect of $|\mathcal{W}| \times |\mathcal{S}|$ As shown in Figure 7, the overall utility grows with the number of workers and the number of tasks for all the four algorithms. DE-Rainbow and DE-DQN outperform 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-Rainbow, 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.

In summary, DE-Rainbow and DE-DQN perform well on both synthetic and real-world datasets. What's more, DE-Rainbow and DE-DQN are 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-Rainbow and DE-DQN can outperform the other comparison algorithms by 30% and 20% respectively.

Ablation study We conduct experiments with only utility embedding $\mathbf{V}^u$ or coverage embedding $\mathbf{V}^c$ and make comparisons with DE-DQN, DE-Rainbow to study the effect of the dual-embedding on DQN and Rainbow. Table 8 shows the comparison with the overall utility of different techniques based on DQN with 1000 and 1200 tasks, while Table 9 shows

Table 8 Comparison based on DQN 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

Table 9 Comparison based on rainbow for 60 workers with capacity mean of 10 on SYN

Number of tasks	$\mathbf{V}^u$+Rainbow	$\mathbf{V}^c$+Rainbow	DE-Rainbow
1000	702.15	763.43	811.31
1200	912.24	883.86	986.53

the comparison based on Rainbow. The results are consistent with the results shown in Figure 5c, 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. In addition, comparing the results of Tables 8 and 9 at the same location shows that Rainbow is superior to the regular DQN. In summary, the results of the ablation study demonstrate that the combination of the Rainbow, utility embedding and coverage embedding is beneficial to the overall performance.

8 Conclusion

In this paper, we focus on solving the global task assignment problem in the spatial crowdsourcing with the goal of maximizing the overall utility for all workers, which is defined as the reward minus the travel cost. We formulate the problem as the Utility-driven Destination-aware Spatial Task Assignment problem (UDSTA) and prove that it is NP-Complete. To deal with the problem, we propose a DE-DQN framework and an enhanced DE-Rainbow framework which balance the current and future utility through the dual-embedding. We construct the proper representations of state, action, and reward of DQN and Rainbow DQN according to our scenario. For the training phase, we use the experience replay buffer to train the Deep Q-Network and use the noisy net to train the Rinbow DQN. The experiments based on both synthetic and real-world datasets verify the effectiveness of the DE-DQN and DE-Rainbow framework. In the future, we will consider the fairness problem where a homogeneity metric is presented to balance the utility among workers.

Acknowledgements Yucen Gao, Dejun Kong and Xiaofeng Gao are from MoE Key Lab of Artificial Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University. Xiaofeng Gao is the corresponding author. The authors want to give special thanks to Wei Liu for her help and contribution to this paper.

Author Contributions Yucen Gao, Dejun Kong, and Xiaofeng Gao wrote the main manuscript text. Haipeng Dai, Xiaofeng Gao, and Jiaqi Zheng suggested some revisions. All authors reviewed the manuscript.

Funding This work was supported by the National Key R&D Program of China [2024YFF0617700, 2023YFB4502400], the National Natural Science Foundation of China [U23A20309, 62272302, 62172276, 62372296, 62272223, U22A2031, 62422207], the Fundamental Research Funds for the Central Universities [2024300349], the Shanghai Municipal Science and Technology Major Project [2021SHZDZX0102], and the CCF-DiDi GAIA Collaborative Research Funds for Young Scholars [202404].

Data Availability The open real-world dataset, Gowalla, could be found at http://snap.stanford.edu/data/loc-Gowalla.html.

Declarations

Ethical Approval Not applicable.

Competing interests The authors declare no competing interests.

References

1. Tong, Y., Zhou, Z., Zeng, Y., Chen, L., Shahabi, C.: Spatial crowdsourcing: a survey. The VLDB Journal (VLDBJ) **29**(1), 217–250 (2020)
2. Dutta, P., Aoki, P.M., Kumar, N., Mainwaring, A., Myers, C., Willett, W., Woodruff, A.: Common sense: participatory urban sensing using a network of handheld air quality monitors. In: ACM Conference on Embedded Networked Sensor Systems (SenSys), pp. 349–350 (2009)
3. 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 (IPSN), pp. 105–116 (2010)
4. Santos, M., Pereira, R.L., Leal, A.B.: Gbus-route geotracer. In: IEEE International Workshop on Vehicular Traffic Management for Smart Cities (VTM), pp. 1–6 (2012)
5. Zhang, X., Gong, H., Xu, Z., Tang, J., Liu, B.: Jam eyes: a traffic jam awareness and observation system using mobile phones. International Journal of Distributed Sensor Networks (IJDSN) **8**(12), 921208 (2012)
6. 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. Strategic Manag. **33**(1), 44–57 (2021)
7. Guo, B., Liu, Y., Wang, L., Li, V.O., Lam, J.C., Yu, Z.: Task allocation in spatial crowdsourcing: current state and future directions. IEEE Internet of Things Journal (IoT-J) **5**(3), 1749–1764 (2018)
8. Duan, L., Zhan, Y., Hu, H., Gong, Y., Wei, J., Zhang, X., Xu, Y.: Efficiently solving the practical vehicle routing problem: A novel joint learning approach. In: ACM Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 3054–3063 (2020)
9. 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 (APWeb-WAIM), pp. 41–56 (2017)
10. Yin, B., Li, J., Wei, X.: Rational task assignment and path planning based on location and task characteristics in mobile crowdsensing. IEEE Transactions on Computational Social Systems (TCSS) **9**(3), 781–793 (2022)
11. Ni, W., Cheng, P., Chen, L., Lin, X.: Task allocation in dependency-aware spatial crowdsourcing. In: IEEE International Conference on Data Engineering (ICDE), pp. 985–996 (2020)
12. Shi, D., Tong, Y., Zhou, Z., Song, B., Lv, W., Yang, Q.: Learning to assign: Towards fair task assignment in large-scale ride hailing. In: ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 3549–3557 (2021)
13. Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M.G., Silver, D.: Rainbow: Combining improvements in deep reinforcement learning. In: AAAI Conference on Artificial Intelligence (AAAI), pp. 3215–3222 (2018)
14. Li, M., Cheng, W., Liu, K., He, Y., Li, X., Liao, X.: Sweep coverage with mobile sensors. IEEE Transactions on Mobile Computing (TMC) **10**(11), 1534–1545 (2011)
15. Bunte, S., Kliewer, N.: An overview on vehicle scheduling models. Public Transport **1**(4), 299–317 (2009)
16. Gao, G., Wu, J., Xiao, M., Chen, G.: Combinatorial multi-armed bandit based unknown worker recruitment in heterogeneous crowdsensing. In: IEEE International Conference on Computer Communications (INFOCOM), pp. 179–188 (2020)
17. 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 (ICDE), pp. 13–24 (2020)
18. Lin, Q., Deng, L., Sun, J., Chen, M.: Optimal demand-aware ride-sharing routing. In: IEEE International Conference on Computer Communications (INFOCOM), pp. 2699–2707 (2018)
19. Zheng, Y., Wang, J., Li, G., Cheng, R., Feng, J.: QASCA: A quality-aware task assignment system for crowdsourcing applications. In: ACM International Conference on Management of Data (SIGMOD), pp. 1031–1046 (2015)
20. Cheng, P., Chen, L., Ye, J.: Cooperation-aware task assignment in spatial crowdsourcing. In: IEEE International Conference on Data Engineering (ICDE), pp. 1442–1453 (2019)
21. Du, Y., Sun, Y.-E., Huang, H., Huang, L., Xu, H., Wu, X.: Quality-aware online task assignment mechanisms using latent topic model. Theoretical Computer Science (TCS) **803**, 130–143 (2020)
22. Cheung, M.H., Hou, F., Huang, J., Southwell, R.: Distributed time-sensitive task selection in mobile crowdsensing. IEEE Transactions on Mobile Computing (TMC) **20**(6), 2172–2185 (2021)
23. Liu, C.H., Zhao, Y., Dai, Z., Yuan, Y., Wang, G., Wu, D., Leung, K.K.: Curiosity-driven energy-efficient worker scheduling in vehicular crowdsourcing: A deep reinforcement learning approach. In: IEEE International Conference on Data Engineering (ICDE), pp. 25–36 (2020)

24. Zhao, Y., Xia, J., Liu, G., Su, H., Lian, D., Shang, S., Zheng, K.: Preference-aware task assignment in spatial crowdsourcing. In: AAAI Conference on Artificial Intelligence (AAAI), pp. 2629–2636 (2019)
25. Wang, X., Jia, R., Tian, X., Gan, X.: Dynamic task assignment in crowdsensing with location awareness and location diversity. In: IEEE International Conference on Computer Communications (INFOCOM), pp. 2420–2428 (2018)
26. Hettiachchi, D., Kostakos, V., Goncalves, J.: A survey on task assignment in crowdsourcing. ACM Comput. Surv. **55**(3), 1–35 (2022)
27. Dai, C., Wang, X., Liu, K., Qi, D., Lin, W., Zhou, P.: Stable task assignment for mobile crowdsensing with budget constraint. IEEE Transactions on Mobile Computing (TMC) **20**(12), 3439–3452 (2020)
28. Zeng, Y., Tong, Y., Chen, L.: Last-mile delivery made practical: An efficient route planning framework with theoretical guarantees. Proceedings of the VLDB Endowment **13**(3), 320–333 (2019)
29. Tong, Y., Zeng, Y., Ding, B., Wang, L., Chen, L.: Two-sided online micro-task assignment in spatial crowdsourcing. IEEE Trans. Knowl. Data Eng. **33**(5), 2295–2309 (2021)
30. Tong, Y., Zeng, Y., Zhou, Z., Chen, L., Xu, K.: Unified route planning for shared mobility: An insertion-based framework. ACM Trans. Database Syst. **47**(1), 2–1248 (2022)
31. Wang, J., Wang, L., Wang, Y., Zhang, D., Kong, L.: Task allocation in mobile crowd sensing: state-of-the-art and future opportunities. IEEE Internet of Things Journal (IoT-J) **5**(5), 3747–3757 (2018)
32. Liu, Q., Peng, J., Ihler, A.T.: Variational inference for crowdsourcing. Advances in Neural Information Processing Systems (NeurIPS) **25** (2012)
33. Wang, H.-n., Liu, N., Zhang, Y.-y., Feng, D.-w., Huang, F., Li, D.-s., Zhang, Y.-m.: Deep reinforcement learning: a survey. Front. Inform. Technol. Electr. Eng. **21**(12), 1726–1744 (2020)
34. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M.A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
35. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems (NeurIPS) **25** (2012)
36. Shan, C., Mamoulis, N., Cheng, R., Li, G., Li, X., Qian, Y.: An end-to-end deep rl framework for task arrangement in crowdsourcing platforms. In: IEEE International Conference on Data Engineering (ICDE), pp. 49–60 (2020)
37. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: International Conference on Learning Representations (ICLR) (2016)
38. Sheng, V.S., Zhang, J.: Machine learning with crowdsourcing: a brief summary of the past research and future directions. In: AAAI Conference on Artificial Intelligence (AAAI), vol. 33, pp. 9837–9843 (2019)
39. Zhao, P., Li, X., Gao, S., Wei, X.: Cooperative task assignment in spatial crowdsourcing via multi-agent deep reinforcement learning. Journal of Systems Architecture: Embedded Software Design (JSA) **128**, 102551 (2022)
40. Wang, Y., Liu, C.H., Piao, C., Yuan, Y., Han, R., Wang, G., Tang, J.: Human-drone collaborative spatial crowdsourcing by memory-augmented and distributed multi-agent deep reinforcement learning. In: IEEE International Conference on Data Engineering (ICDE), pp. 459–471 (2022)
41. Wang, H., Liu, C.H., Dai, Z., Tang, J., Wang, G.: Energy-efficient 3d vehicular crowdsourcing for disaster response by distributed deep reinforcement learning. In: ACM Conference on Knowledge Discovery & Data Mining (SIGKDD), pp. 3679–3687 (2021)
42. Sun, Y., Liu, M., Huang, L., Xie, N., Zhao, L., Tan, W.: An embedding-based deterministic policy gradient model for spatial crowdsourcing applications. In: International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 1268–1274 (2021)
43. Shan, C., Mamoulis, N., Cheng, R., Li, G., Li, X., Qian, Y.: An end-to-end deep RL framework for task arrangement in crowdsourcing platforms. In: International Conference on Data Engineering (ICDE), pp. 49–60 (2020)
44. Ye, G., Zhao, Y., Chen, X., Zheng, K.: Task allocation with geographic partition in spatial crowdsourcing. In: ACM International Conference on Information & Knowledge Management (CIKM), pp. 2404–2413 (2021)
45. 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 (CIKM), pp. 1355–1364 (2020)
46. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning (ICML), pp. 1928–1937 (2016)

47. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International Conference on Machine Learning (ICML), pp. 1889–1897 (2015)
48. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv:1707.06347 (2017)
49. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv:1312.5602 (2013)
50. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: International Conference on Machine Learning (ICML), pp. 387–395 (2014)
51. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning (ICML), pp. 1587–1596 (2018)
52. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning (ICML), pp. 1861–1870 (2018)
53. Racanière, S., Weber, T., Reichert, D., Buesing, L., Guez, A., Jimenez Rezende, D., Puigdomènech Badia, A., Vinyals, O., Heess, N., Li, Y., et al.: Imagination-augmented agents for deep reinforcement learning. Advances in neural information processing systems (NeurIPS) **30** (2017)
54. Bansal, S., Calandra, R., Chua, K., Levine, S., Tomlin, C.: Mbmf: Model-based priors for model-free reinforcement learning. arXiv:1709.03153 (2017)
55. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
56. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: AAAI Conference on Artificial Intelligence (AAAI), pp. 2094–2100 (2016)
57. Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N.: Dueling network architectures for deep reinforcement learning. In: International Conference on Machine Learning (ICML), pp. 1995–2003 (2016)
58. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. In: International Conference on Learning Representations (ICLR) (2016)
59. Fortunato, M., Azar, M.G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., Legg, S.: Noisy networks for exploration. In: International Conference on Learning Representations (ICLR) (2018)
60. Bellemare, M.G., Dabney, W., Munos, R.: A distributional perspective on reinforcement learning. In: International Conference on Machine Learning (ICML), vol. 70, pp. 449–458 (2017)
61. Zheng, L., Cheng, P., Chen, L.: Auction-based order dispatch and pricing in ridesharing. In: IEEE International Conference on Data Engineering (ICDE), pp. 1034–1045 (2019)
62. Gutin, G., Punnen, A.P. (eds.): The Traveling Salesman Problem and Its Variations, (2007)
63. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 1082–1090 (2011)
64. Liu, W., Gao, X.: Leveraging social networks to enhance effective coverage for mobile crowdsensing. In: IEEE International Conference on Web Services (ICWS), pp. 389–393 (2020)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Yucen Gao[1] · Dejun Kong[1] · Haipeng Dai[2] · Xiaofeng Gao[1] · Jiaqi Zheng[2] · Fan Wu[1] · Guihai Chen[2]

✉ Xiaofeng Gao
gao-xf@cs.sjtu.edu.cn

Yucen Gao
guo_ke@sjtu.edu.cn

Dejun Kong
kdjkdjkdj99@sjtu.edu.cn

Haipeng Dai
haipengdai@nju.edu.cn

Jiaqi Zheng
jzheng@nju.edu.cn

Fan Wu
fwu@cs.sjtu.edu.cn

Guihai Chen
gchen@nju.edu.cn

1 Shanghai Jiao Tong University, Shanghai 200240, China

2 Nanjing University, Nanjing 210008, China