Curiosity-Driven Energy-Efficient Worker Scheduling in Vehicular Crowdsourcing: A Deep Reinforcement Learning Approach

Chi Harold Liu
School of Comp. Sci. and Tech.
Beijing Inst. of Tech.
Beijing, China
chiliu@bit.edu.cn

Yinuo Zhao
School of Comp. Sci. and Tech.
Beijing Inst. of Tech.
Beijing, China
3120191078@bit.edu.cn

Zipeng Dai
School of Comp. Sci. and Tech.
Beijing Inst. of Tech.
Beijing, China
3120190984@bit.edu.cn

Ye Yuan
School of Comp. Sci. and Tech.
Beijing Inst. of Tech.
Beijing, China
yuan-ye@bit.edu.cn

Guoren Wang
School of Comp. Sci. and Tech.
Beijing Inst. of Tech.
Beijing, China
wanggr@bit.edu.cn

Dapeng Wu
ECE Dept.
University of Florida
Gainesville, USA
wu@ece.ufl.edu

Kin K. Leung
EEE and Computing Dept.
Imperial College
London, U.K.
kin.leung@imperial.ac.uk

***Abstract*—Spatial crowdsourcing (SC) utilizes the potential of a crowd to accomplish certain location based tasks. Although worker scheduling has been well studied recently, most existing works only focus on the static deployment of workers but ignore their temporal movement continuity. In this paper, we explicitly consider the use of unmanned vehicular workers, e.g., drones and driverless cars, which are more controllable and can be deployed in remote or dangerous areas to carry on long-term and hash tasks as a vehicular crowdsourcing (VC) campaign. We propose a novel deep reinforcement learning (DRL) approach for curiosity-driven energy-efficient worker scheduling, called "DRL-CEWS", to achieve an optimal trade-off between maximizing the collected amount of data and coverage fairness, and minimizing the overall energy consumption of workers. Specifically, we first utilize a chief-employee distributed computational architecture to stabilize and facilitate the training process. Then, we propose a spatial curiosity model with a sparse reward mechanism to help derive the optimal policy in large crowdsensing space with unevenly distributed data. Extensive simulation results show that DRL-CEWS outperforms the state-of-the-art methods and baselines, and we also visualize the benefits curiosity model brings and show the impact of two hyperparameters.**

***Index Terms*—Vehicular crowdsourcing, deep reinforcement learning, worker scheduling, curiosity model**

I. Introduction

Spatial crowdsourcing (SC [1]) offers a way to utilize the wisdom of a crowd to perform certain tasks that need on-location operations. In a SC system, a server assigns a task to a batch of workers, under some constraints, such as travel budget, task deadline, and limited transmission bandwidth. A more specific scenario in SC is mobile crowd sensing (MCS), where workers use their carried smart devices to perform sensing tasks, like taking photos, shooting a video, etc. These sensing data are then uploaded to the server, in order to provide a specific service to its users. For example, Google Maps provide street view service by encouraging its users to share their nearby scenery. Certain locations, where street view car cannot access, like campus and museums, are available in it through user contributions by MCS.

Drones and driverless cars offer new opportunities for SC and MCS, as a vehicular crowdsourcing (VC) campaign. They are equipped with high precision sensors to perform the data collection tasks from point-of-interests (PoIs) in remote or dangerous areas, like earthquake rescue, traffic monitoring, or disaster reporting. Moreover, drones can be recharged by landing on the charging station and taking back off again, which makes a long-term task performance possible. In this paper, we explicitly consider the problem of worker scheduling in VC, to maximize the overall collected sensing data under constrained energy budget. Under this scenario, the first challenge is that to derive a long-term optimal policy for worker scheduling could be almost impossible due to its spatiotemporal data complexity and correlation. Second, with the presence of multiple randomly distributed charging stations, when and where to charge becomes an issue, i.e., finding a trade-off between task performance and energy charging is non-trivial. Finally, sensing tasks/data are unevenly distributed. For example, the earthquake could result in different degree of damage of different locations.

Recent research progress in the field of deep reinforcement learning (DRL) achieves great success to derive optimal policy in large exploration space and outperforms humans on several complex decision-making problems, by using powerful deep neural networks (DNNs) for model representation [2]. Our contribution in this paper is three-fold:

1) We propose a DRL-based solution called "DRL-CEWS" to maximize the long-term overall collected data and obtain a trade-off between data collection and energy

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00010

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.
IEEE computer society

TABLE I: List of important notations used in this paper.

Notation	Explanation
w, W	Index of an intelligent worker, total no. of workers
p, P	Index of a PoI, total # of PoIs
t, T	Timeslot, total # of timeslots
$q_t^w, b_t^w, e_t^w, \sigma_t^w$	Data collection, energy budget, energy comsumption, charged energy value of worker w at timeslot t
δ_t^p	Data value for a PoI p at timeslot t
d	Euclidean distance function
$r_t^{int}, r_t^{ext}, r_t$	Intrinsic, extrinsic and total reward at timeslot t
u_t^w, v_t^w	Charging and route planning decision for w at timeslot t

charging in VC. Specifically, we utilize a distributed chief-employee computational architecture to stabilize and facilitate the policy optimization process; and we use a convolutional neural network (CNN) to strengthen the capacity of extracting the spatial features in our crowdsensing space.

2) We propose a spatial curiosity model to predict the workers' future locations and take the prediction error as an intrinsic reward in complex scenarios like areas with unevenly distributed data. We further design a simple and general sparse reward mechanism, to derive a more efficient policy.
3) We conduct extensive experiments for feature, visualize the curiosity effect, and verify the effectiveness of our proposed solution while comparing with four other approaches.

The rest of the paper is organized as follows. First, we review related works in Section II. Then, we present the system model and problem formulation in Section III and necessary background information in Section IV. Next, we propose our proposed solution DRL-CEWS in Section V and present the corresponding pseudocode in Section VI. Finally, we show the performance evaluation in Section VII, and draw the conclusion in Section VIII.

II. Related Work

A. Spatial Crowdsourcing (SC)

SC attempts to match participants with the spatial-temporal tasks according to their availability and constraints. SC has been applied in many kinds of systems including real time traffic speed estimation [3, 4], online car hailing service [5], data labeling [6], etc. Existing studies can be categorized in algorithmic model, which can be matching or planning [1]. In the matching model, algorithms aimed at deriving an optimal task assignment policy, i.e., assigning a task to a worker, with different goals. For example, in order to improve crowdwork quality, Lian *et al.* in [7] considered overall data quality, and proposed a prediction-based solution to find optimal task-worker pair. Chen *et al.* in [5] considered the problem as minimizing maximum delay in dynamic worker and requester system, and proposed a space embedding based online random algorithm to solve it. Wang *et al.* in [8] proposed a Restricted Q-learning method to solve the online matching problem where both tasks and workers arrive dynamically. Tong *et al.* in [9] focused on an online micro-task allocation problem and proposed a TGOA-Greedy algorithm to perform a global optimal match. In the planning model, task assignment aims to plan a route for each worker to perform a sequence of tasks, considering the deadline and travel cost of tasks.

B. Mobile Crowd Sensing (MCS)

MCS can be considered as a special application of SC, when workers are assigned tasks of sensing data collection, like collecting CCTV camera data [10]. Zhou *et al.* in [11] considered using drones, but ignore the energy charging issue, which cannot achieve long time monitoring services in practice. Liu *et al.* in [12] employed mobile vehicles for data collection and sensor node charging. Considering most SC problems have a scalar metric to optimize (e.g., quality, cost and latency) [13], and traditional method cannot achieve a long-term optimization solution, DRL based methods are getting more attention in SC and MCS problems. Wang *et al.* in [14] proposed a DRL-based vehicle selection algorithm to maximize spatial temporal coverage in MCS. Abedin *et al.* in [15] proposed the energy efficient intelligent crowdsourcing during natural calamity through UAVs. Our goal is to design a novel DRL based solution to make navigation decisions for VC, with the presence of multiple charging stations and randomly distributed PoIs.

III. System Model and Problem Formulation

In this section, we first present our system model. Then we formulate our problem as an Optimal Long-term Data Collection (OLDC) problem. Finally, we discuss the metrics we used to evaluate the performance of our algorithm and other trajectory planning algorithms.

A. System Model

Definition 1. ***(Crowdsensing Space)*** *A crowdsensing space is a 2-D metric space $\mathcal{L} = \{(x, y) \,|0 < x < L_x, 0 < y < L_y.\}$, where L_x and L_y are the maximum length along with x, y, respectively. For any two positions $(x_i, y_i), (x_j, y_j)$ inside the space, the distance function $d(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ indicates the Euclidean distance between their locations.*

With respect to (w.r.t.) most cases in the real world, a worker's traveling distance has a fixed maximum given a discretized time slot. Without loss of generality, we assume a sensing task is deployed in the crowdsensing space, where intelligent workers, PoIs, charging stations, obstacles are distributed.

Definition 2. ***(Intelligent Workers)*** *Let $\mathcal{W} = \{1, 2, ..., W\}$ be a set of W autonomous moving workers in the crowdsensing space, e.g., drones and driverless cars. Let (x_t^w, y_t^w) be the current coordinate of worker w at time t, and each w has its own energy budget b_t^w. When $b_t^w = 0$, the worker w stops movement. Thus the intelligent workers need to travel to charging stations before $b_t^w = 0$. Let g^w be the sensing*

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.

range for worker w, which represents the maximum sensing capability in terms of PoI coverage, e.g., shooting range or facing direction of a camera.

Definition 3. ***(PoIs)*** *PoIs refer to a set of points which contain valuable information (e.g., data points or streaming videos) to be obtained by intelligent workers in the crowdsensing space, defined as $\mathcal{P} = \{1, 2, ..., P\}$. For each PoI p, it locates at coordinate (x^p, y^p) and has a value of δ_t^p at time slot t. In practice, PoIs are unevenly distributed.*

We consider the data associated with each PoI have different value, e.g., different cameras captures images of diverse spatial resolutions, different audio life detection instruments generate data of different qualities. An intelligent worker can wait to be charged if and only if: (a) the traveling distance between the worker and charging station is less than a specific range; and (b) the worker receives a charging decision from the server. We define the obstacles as regions which workers cannot enter or go through, e.g., buildings or any ongoing engineering work.

In this paper, we explicitly solve the worker scheduling problem which refers to the route planning of a group of intelligent workers in the crowdsensing space given the distributed PoIs, obstacles and charging stations. We formulate it as an OLDC problem, where the ultimate goal is to maximize long-term overall collected amount of data, coverage fairness among PoIs, under limited worker energy budget. The overall process is described as follows. We have W workers in our crowdsensing space. At each time slot t, the server makes valid navigation decision for each worker w. After the worker w receives the command, it travels to the position $\left(x_{t+1}^w, y_{t+1}^w\right)$ from (x_t^w, y_t^w), either collects data from PoIs within the sensing range g^w from the center position $\left(x_{t+1}^w, y_{t+1}^w\right)$ or charges itself. We denote the current collected amount of data by worker w as:

$$q_t^w = \sum_{p=1}^{P} \chi_w(p) \min(\lambda \delta_0^p, \delta_t^p), \tag{1}$$

where λ denotes the data collection rate, which models the practical scenario that a worker may not obtain all data at once, with the limited sensing time in a time slot, e.g., due to shooting angles/directions, limited field of view, or streaming nature of video data. χ is a mapping function that indicates whether a PoI p is within the worker w's sensing range, as:

$$\chi_w(p) = \begin{cases} 1, & \text{if } d\left[\left(x_{t+1}^w, y_{t+1}^w\right), (x^p, y^p)\right] \leqslant g^w, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Thus, the overall collected data by worker w up to time slot t is denoted by $Q_t^w = \sum_{k=1}^{t} q_k^w$.

Next, we define the energy consumption model. Let e_t^w be the consumed amount of energy by worker w at time slot t, which is formulated as:

$$e_t^w = \beta * d[(x_t^w, y_t^w), (x_{t+1}^w, y_{t+1}^w)] + \alpha * q_t^w, \tag{3}$$

where the first part is the traveling cost, and second part is incurred by the data collection, where α is the energy consumption per unit of data collected and β is the energy consumption per unit of traveling distance. Thus, the current energy budget of worker w is denoted as $b_t^w = b_{t-1}^w - e_t^w + \sigma_t^w$, where σ_t^w donates charged energy value of worker w at timeslot t. The total energy consumption by worker w up to time slot t is $E_t^w = \sum_{k=1}^{t} e_k^w$.

B. Evaluation Metrics

In order to evaluate the policy performance and define the objectives of OLDC problem, we introduce three metrics.

Definition 4. ***(Average Data Collection Ratio)*** *It is defined as the averaged ratio between total collected data and inital data amount for all workers up to time slot t, as:*

$$\kappa_t = \frac{1}{W} \frac{\sum_{w=1}^{W} Q_t^w}{\sum_{p=1}^{P} \delta_0^p}. \tag{4}$$

Definition 5. ***(Average Remaining Data Ratio)*** *Since any single worker cannot collect all PoIs at a time slot, it is important to measure the coverage fairness ξ_t geographically, which is calculated by the average remaining data ratio for all PoIs:*

$$\xi_t = \frac{1}{P} \sum_{p=1}^{P} \frac{\delta_t^p}{\delta_0^p}. \tag{5}$$

Definition 6. ***(Energy Efficiency)*** *We define a spatiotemporal metric "energy efficiency" ρ_t, taking the input of collected data amount Q_t^w and energy consumption E_t^w, weighted by a Jain's fairness index [16], among all workers, as:*

$$\rho_t = \frac{\left(\sum_{p=1}^{P} \left\lceil \frac{\delta_0^p - \delta_t^p}{\lambda \delta_0^p} \right\rceil\right)^2}{P \sum_{p=1}^{P} \left(\left\lceil \frac{\delta_0^p - \delta_t^p}{\lambda \delta_0^p} \right\rceil\right)^2} \frac{1}{W} \sum_{w=1}^{W} \frac{Q_t^w}{E_t^w}. \tag{6}$$

For a task duration T time slots, an optimal worker scheduling policy π^* for OLDC problem is defined as:

$$\pi^* = \arg\max_{\pi} \rho_T. \tag{7}$$

We observe four factors in practice, that make OLDC problem difficult to solve. First, data associated with PoIs are scattered unevenly in the crowdsensing space, which makes the optimal policy difficult to derive under different circumstances. Second, obstacles in our condition may largely weaken the worker's exploration capability. In rescue cases workers like drones should have to go into some corner collapse areas for life search. Third, workers need to learn to be cooperative (each should be responsible for a subarea not all of the space) and be competitive (number of charging stations in practice is not enough for all workers simultaneously, and easily found PoIs need to be covered only by a small number of workers). Fourth, when and where to charge the battery becomes an issue since replenishing battery will increase the future chance of task completion, but it takes time that workers cannot collect data at the current time slots.

Obviously, our OLDC problem is NP-hard, and thus intractable. Alternatively, we seek to design an efficient heuristic method to tackle the OLDC problem, based on DRL.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.

IV. PRELIMINARIES

Reinforcement learning (RL) is to learn a mapping function, from state $\boldsymbol{s}$ to action $\boldsymbol{a}$, so as to maximize the reward r. Recently, DRL has developed quickly and has been successfully applied to computer games [2], [17] and so on. Policy gradient algorithms in DRL aim at getting an optimal policy network which maximizes the total sum of reward, i.e., $J(\theta) = \mathbb{E}[\sum_{t=1}^{T} \gamma^{t-1} r(\boldsymbol{s}_t, \boldsymbol{a}_t)]$, where T is the end time.

In order to measure the relative advantages between different actions, it is common to use the mean value estimate c_t across the accessible action space, which does not depend on current actions or update quickly. Thus, the objective turns to $J(\theta) = \mathbb{E}[\sum_{t=1}^{T} \gamma^{t-1}(r(\boldsymbol{s}_t, \boldsymbol{a}_t) - c_t)]$. For $r(\boldsymbol{s}_t, \boldsymbol{a}_t) - c_t$, we use A_t instead, which indicates how better $\boldsymbol{a}_t$ is, compared to c_t. Policy gradient methods are popular in DRL, as they have better converging performance and more efficient in high dimensional action space, compared with other methods, i.e. value gradient ones. However, policy gradient methods have some drawbacks, like high variance and inefficient training process. In [17], Schulman *et al.* proposed a robust and simple policy gradient algorithm, called proximal policy gradient (PPO). It uses a clipped objective to constraint the updating step in a trust region, as:

$$J(\theta) = \hat{\mathbb{E}}_t[\min(\zeta_t(\theta), CLIP(\zeta_t(\theta), 1-\epsilon, 1+\epsilon))\hat{A}_t], \quad (8)$$

where $\zeta_t(\theta) = \frac{\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{\theta_{old}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}$ denotes the probability ratio with respect to old policy $\pi_{\theta_{old}}$ and updated policy π_θ. The $CLIP$ term removes the incentive for moving $\zeta_t(\theta)$ out of interval $[1-\epsilon, 1+\epsilon]$, which makes the training process more stable. In A_t, they use the state value $V(\boldsymbol{s}_t)$ to be the baseline.

PPO is relatively simple to implement, more robust and has a high generality.

V. PROPOSED SOLUTION: DRL-CEWS

In this section, we first model our task assignment as a Markov Decision Process (MDP), which is the fundamental assumption of any DRL solution. Then, we introduce our distributed computational architecture and DNN model design. Finally, we present our sparse reward mechanism design.

An MDP is a discrete time stochastic control process, which is useful for studying optimization problems solved by RL methods. In our worker scheduling problem, the server decides the optimal policy for each worker, and the decision-making process of worker navigation only depends on current state, i.e., the current information in a crowdsensing space, which satisfies the Markov property.

State: Let $\boldsymbol{s}_t$ define the state at time slot t, which is formulated as a matrix of 3 channels. In the first channel, we place the energy budget b_t^w for each worker w at its current position (x_t^w, y_t^w). In the second channel, we place the position and PoI data into the matrix, including the current position for charging station and obstacles, the current remaining data value δ_t^p at each PoI p. In the third channel, we place the PoIs' access time $h_t(p) \in (0, T]$ on their location. That is, if a PoI p is sensed and its data is collected in time slot $t+1$, we have $h_{t+1}(p) = h_t(p) + 1$. We intentionally include this in our state, to make sure the server is aware of the coverage fairness among all PoIs.

Action: There are two kinds of actions for a worker: the energy charging decision $\boldsymbol{u}_t$ and the route planning decision $\boldsymbol{v}_t$. Let $\boldsymbol{u}_t = \{u_t^1, u_t^2, ..., u_t^W\}$ which indicates whether a worker should charge itself now or not, where u_t^w specified the decision for worker w at the current location (x_t^w, y_t^w). It is valid for a worker to charge when the traveling distance between worker and charging station is less than a specific range. On the other hand, the route planning decision at time slot t is denoted as $\boldsymbol{v}_t$, which indicates the valid next position (x_{t+1}^w, y_{t+1}^w) a worker w should move to. Specifically, $\boldsymbol{v}_t = \{v_t^1, v_t^2, \ldots, v_t^w\}$, where $x_{t+1}^w = x_t^w + x(v_t^w)$, $y_{t+1}^w = y_t^w + y(v_t^w)$ for worker w at time slot t. A valid route planning action for a worker w is when: (a) w will not bump into the obstacles or go beyond the crowdsensing; (b) the current energy budget b_t^w is not exhausted; (c) $\|v_t\|_2$ can not exceed a fixed maximum of a worker's traveling distance. Therefore, the whole action $\boldsymbol{a}_t$ is defined as:

$$\boldsymbol{a}_t = [\boldsymbol{u}_t, \boldsymbol{v}_t] \quad (9)$$

Reward: We consider both extrinsic reward and intrinsic reward in our solution. The former is human designed, while the latter is independent of environment and generated by the DRL agent itself. The total reward in our system is defined as the sum of intrinsic reward r_t^{int} and extrinsic reward r_t^{ext}:

$$r_t = r_t^{int} + r_t^{ext}. \quad (10)$$

The intrinsic reward will be explained in Section V-C, and the sparse extrinsic reward mechanism will be explained in Section V-D. Although our solution does not need dense extrinsic reward, in order to train other DRL-based baselines, i.e., Edics and DPPO, we give the dense extrinsic reward definition in Section VII-B.

In an MDP, we can easily transfer our OLDC from a NP-hard problem to a series of decision-making problems. For DRL methods running with powerful DNNs, the exploration and exploitation process can be improved and accelerated significantly, compared to many traditional methods such as dynamic programming and genetic algorithm.

We next introduce our overall model in three steps, as is shown in Fig. 1. First is the distributed computational architecture, which consists of one chief thread and multiple employee threads. Second is the spatial curiosity model, which generates the intrinsic reward. Third is the sparse reward mechanism where we will explain how intrinsic reward works.

A. Chief-Employee Distributed Computational Architecture

As shown in Fig. 1, it consists of two parts: a chief thread and multiple employee threads, with PPO as the policy network. Although asynchronous setting can be more efficient than the synchronous one, the decoupling between data sampling and policy learning will result in a *policy-lag* [18] between chief and employees, which will further

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.

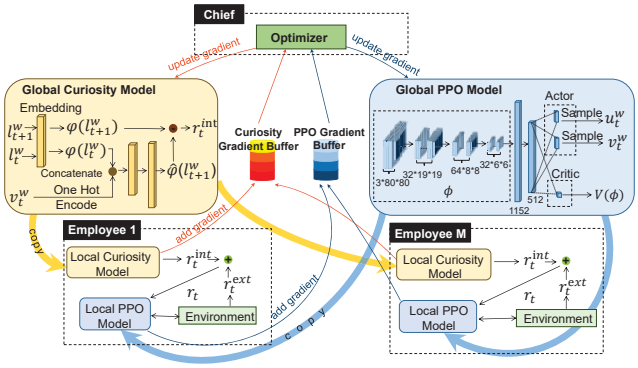


Fig. 1: Overall proposed DNN model.

make the learning process unstable. Espeholt *et al.* in [18] proposed a novel off-policy correction method called V-trace to achieve stable learning in single-players games, such as Atari. However, the number of drones in practice will be large. Therefore, in our setting, we simply adopt a synchronous structure to ensure robustness and stability in most cases. The chief thread works as follows. First, it waits until all the employees have sent the gradient w.r.t. PPO network parameters. Second, the optimizer in chief updates the global PPO model parameters along the negative direction of the gradient. Meanwhile, each employee thread works in parallel and has independent local PPO model and local environment. In Fig. 1, we take Employee 1 for example. First, it copies the parameters from the global PPO model into its local PPO model. Then, it interacts with the local environment using the policy from local PPO and stores the experiences. Third, it uses the stored experiences to compute the loss function. Finally, it derives the gradient w.r.t. local PPO model parameters and sends the gradient to PPO gradient buffer.

There are two global gradient buffers in our architecture, i.e., PPO gradient buffer and curiosity gradient buffer, as shown in the center part of Fig. 1. The PPO gradient buffer accepts the gradient sent by employee threads from local PPO model, sums them up, and sends them to chief. The curiosity gradient buffer works the same and serves for spatial curiosity model. In this way, the proposed Chief-Employee distributed computational architecture can help generate much more diverse experiences and to make the update process more stable and quicker to converge.

B. DRL Agent by PPO for Multi-Worker Action Generations

As introduced in Section IV, PPO can derive optimal and robust policy by updating steadily constraint by a clip function. Considering the strong spatial characteristics of workers, obstacles, charging stations, PoIs, and the relationship between them, we use the powerful CNN to extract the feature of the state. Given the state in our system is not as complicated as a real image, we adopt a small CNN which consists of three convolutional layers and one fully collected layers to output a 1D state feature $\phi(\boldsymbol{s}_t)$. We add layer normalization to make the updating process more stable, as shown the dark blue layer after each CNN layer in Fig. 1.

The value network V aims at predicting the accumulated reward obtained by the workers from time slot t to the task completion time slot T. The input is $\phi(\boldsymbol{s}_t)$ and the output is a scalar value V. We update the V by minimizing the loss function as:

$$Loss^v = (\|G_t, V(\phi(\boldsymbol{s}_t))\|_2)^2, \tag{11}$$

where $G_t = r_t + \gamma r_{t+1} + ... + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T)$, which denotes the expected sum of reward till time slot T.

For the policy network π, we use the clip surrogate objective to derive the energy charging action and movement action for each worker, as:

$$J = \hat{\mathbb{E}}_t[\min(\zeta_t \hat{A}_t, CLIP(\zeta_t, 1-\epsilon, 1+\epsilon)\hat{A}_t)]. \tag{12}$$

C. Spatial Curiosity Model for Generating Intrinsic Reward

A good reward design is key for a successful policy learning. DRL algorithms work well on game problems, whose goal is quite explicit and the reward is densely distributed. However, in many real life scenarios, densely distributed reward is hard to obtain and the DRL agent may fail to derive an optimal policy. Humans, on the other hand, show a strong learning ability in such scenarios. For example, on your first day in school, you walk around the campus and get yourself familiar with the strange environment, without any explicit objectives. Mimicking human behaviors, a DRL agent generates its own *intrinsic* reward, which we call "curiosity", to accelerate its training process. Pathak *et al.* in [19] proposed his curiosity model consisting of three networks. The first is an encoding network, mapping current state $\boldsymbol{s}_t$ to its feature $\varphi(\boldsymbol{s})$, next state $\boldsymbol{s}_{t+1}$ to $\varphi(\boldsymbol{s}_{t+1})$. This feature network extracts the spatial feature and focuses on the changes in environment that only due to the agent's action. The second is a forward model, that takes inputs $\varphi(\boldsymbol{s}_t)$ and $\boldsymbol{a}_t$, and predicts the feature encoding of the state at time slot $t+1$, as:

$$\hat{\varphi}(\boldsymbol{s}_{t+1}) = f(\varphi(\boldsymbol{s}_t), \boldsymbol{a}_t; \theta_F), \tag{13}$$

where $\hat{\varphi}(\boldsymbol{s}_{t+1})$ is the predicted feature encoding of $\boldsymbol{s}_{t+1}$. The forward model aims at minimizing the loss function:

$$Loss^f(\hat{\varphi}(\boldsymbol{s}_{t+1}), \varphi(\boldsymbol{s}_{t+1})) = (\|\hat{\varphi}(\boldsymbol{s}_{t+1}) - \varphi(\boldsymbol{s}_{t+1})\|_2)^2. \tag{14}$$

Meanwhile, the forward model loss $Loss^f$ applies to the intrinsic reward r_t^{int} at time slot t, i.e., $r_t^{int} = \eta Loss^f$, where η is a scaling factor to control the proportion of intrinsic reward in total sum of extrinsic and intrinsic rewards. In order to make the feature encoding network φ more reliable, an inverse model is used to predict the action $\boldsymbol{a}_t$, given $\varphi(\boldsymbol{s}_t)$ and $\varphi(\boldsymbol{s}_{t+1})$.

In this paper, we propose a spatial curiosity model. We formulate curiosity as the continuous exploration for a novel state, including: (1) Traveling to where workers have seldom or even never visited before; (2) Taking different route planning $\boldsymbol{v}_t$ at the same location; (3) Avoiding frequent collisions with obstacles in the early exploration process. More specifically, we predict the embedding feature $\hat{\varphi}(\boldsymbol{l}_{t+1})$ of worker's next position, given each worker's current position, denoted as

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.

$l_t^w = (x_t^w, y_t^w)$ and route planning decision v_t^w in a forward model, denoted as f, where:

$$\hat{\varphi}(\boldsymbol{l}_{t+1}) = f(\varphi(\boldsymbol{l}_t), \boldsymbol{v}_t). \tag{15}$$

We update the forward model by minimizing the loss function:

$$Loss^f = (\|\hat{\varphi}(\boldsymbol{l}_{t+1}) - \varphi(\boldsymbol{l}_{t+1})\|_2)^2. \tag{16}$$

As mentioned, the intrinsic reward r_t^{int} is computed by:

$$r_t^{int} = \eta Loss^f. \tag{17}$$

D. Sparse Reward Mechanism

We design a sparse reward mechanism as the extrinsic reward for our solution. The intuition behind is to reward agent intermittently, so as to avoid being trapped in a local suboptimal policy. Dense reward is essential in DRL, where the agent gets a reward every time slot, and updates its policy according to the accumulated reward. However, dense reward could also lead to a local sub-optimal policy optimization. For example, in a crowdsensing space where PoIs are unevenly distributed, a worker may get higher reward in a rich data area, but get much lower reward in sparse data area. Apart from the data importance in sparse area, in this case, the worker may lose the opportunity to go through the sparse data area and fail to find charging stations due to limited energy budget. Therefore. we should design a mechanism to reward the agent for its long-term data collection, rather than focus on present.

We propose a *simple* and *sparse* extrinsic reward, where the DRL agent gets reward intermittently from time to time, as:

$$r_t^{w,ext} = \Upsilon_t^1 + \Upsilon_t^2 - \tau_t^w, \tag{18}$$

where $\Upsilon_t^1 = 1$, whenever the data collection ratio κ_t^w increases ϵ_1, otherwise 0; $\Upsilon_t^2 = 1$, whenever the charged energy ratio σ_t^w / b_0^w is larger than ϵ_2, otherwise 0. Here ϵ_1 and ϵ_2 are two bounds as hyperparameters. τ_t^w represents the penalty given to worker w if it hits the obstacle. Then, our total extrinsic reward r_t^{ext} is formulated as:

$$r_t^{ext} = \frac{1}{W} \sum_{w=1}^{W} r_t^{w,ext}. \tag{19}$$

VI. Algorithm Description

We implement our algorithm in a synchronous manner, composed of one chief thread and multiple employee threads. The former is responsible for collecting the gradient sending from the employee threads and then updating the global model using an optimizer, e.g., Adam optimizer. The employee threads sample actions from the model, interact with environment, compute and send parameter gradients to the chief.

A. Pseudocode in Algorithm 1: Employee threads

The input is the current state $\boldsymbol{s}_t$, while the output is policy $\boldsymbol{u}_t$ and $\boldsymbol{v}_t$ for every worker. We first randomly initialize the state $\boldsymbol{s}_0$ and the parameters of embedding feature extractor φ, policy network π, value network V, forward model f, CNN model ϕ, along with the empty replay buffer $\mathcal{D}$ to store the historical information (Line 1). For each episode, there are exploration (Line 4-15) and exploitation (Line 17-24).

Algorithm 1 Pseudocode: employees

Input: current state $\boldsymbol{s}_t$
Output: energy charging action $\boldsymbol{u}_t$ and route planning action $\boldsymbol{v}_t$

1: Initialize $\boldsymbol{s}_0$, ϕ, π, V, f, φ and replay buffer $\mathcal{D}$;
2: **for** Episode in 1, 2, $\cdots$ **do**
3: Clear up replay buffer $\mathcal{D}$
4: **for** t in 1, 2, $\cdots$, T **do**
5: Get feature representation $\phi(\boldsymbol{s_t})$ from CNN;
6: Derive route planning action $\boldsymbol{v}_t$ and energy charging action $\boldsymbol{u}_t$ from policy model π;
7: **for** w in 1, 2, $\cdots$, W **do**
8: broadcast u_t^w and v_t^w to worker w;
9: Receive the overall collected data and next position from worker w;
10: **end for**
11: Calculate the sparse extrinsic reward r_t^{ext} through Eqn. (18) and Eqn. (19);
12: Derive the intrinsic reward r_t^{int} through Algorithm 3;
13: Calculate the total reward through Eqn. (10);
14: Update the replay memory buffer $\mathcal{D}$ by adding a new record $[\boldsymbol{s}_t, \boldsymbol{u}_t, \boldsymbol{v}_t, r_t]$;
15: **end for**
16: $\pi_{old} \leftarrow \pi$;
17: **for** k in 1, 2, ... K **do**
18: Sample a mini batch experience from $\mathcal{D}$;
19: Compute gradients w.r.t π, V, ϕ and f;
20: Send gradients to chief;
21: Wait until notification from chief;
22: Copy parameters from global PPO and curiosity model to the employee's local model, respectively;
23: **end for**
24: **end for**

For exploration, the algorithm collects history samples. At t, we first feed $\boldsymbol{s}_t$ into CNN and extract the efficient feature $\phi(\boldsymbol{s}_t)$. Then, we feed $\phi(\boldsymbol{s}_t)$ into the policy model π and derive the route planning policy probability distribution across action space, and sample action vector $\boldsymbol{v}_t$ from it (Line 6). Also, we derive the charging policy probability distribution from π and sample energy charging action vector $\boldsymbol{u}_t$ from it (Line 6). After obtaining the policy, the server notifies every worker its action u_t^w, v_t^w (Line 8). After that, each worker moves accordingly. Next, each worker reports the sensing data and its own status (e.g., remaining energy, current location) back to the server, the server calculates the sparse extrinsic reward by Eqn. (19) (Line 11). We obtain the intrinsic reward from Algorithm 3 (Line 12) and combine it with r_t^{ext}. Finally, the employee inserts a new experience into the memory buffer $\mathcal{D}$. This process repeats and ends at time slot T.

After T time slots, each employee starts exploitation to update policy network π, value network V, CNN ϕ, forward network f. For each round of update k (Line 17), the employee

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2 Pseudocode: chief

1: **for** Episode in 1, 2, $\cdots$ **do**
2: **for** k in 1, $\cdots$, K **do**
3: Wait until all employees son have sent gradients w.r.t π, V and ϕ;
4: Update global PPO model;
5: Wait until all the son have sent gradients w.r.t f;
6: Update global spatial curiosity model;
7: Clear global gradient buffers;
8: Notify all employees;
9: **end for**
10: **end for**

Algorithm 3 Pseudocode: Spatial Curiosity Model

Input: workers position (x_t^w, y_t^w), (x_{t+1}^w, y_{t+1}^w)
Output: intrinsic reward r_t^{int}

1: **for** w in 1, 2, ... W **do**
2: Get embedding feature φ of position (x_t^w, y_t^w), (x_{t+1}^w, y_{t+1}^w) respectively;
3: **end for**
4: Get $Loss^f$ in Eqn. (16)
5: Get intrinsic reward in Eqn. (17)

first samples a mini batch of experiences from $\mathcal{D}$. Then, it calculates J by Eqn. (12), and uses back propagation to calculate the gradient w.r.t. network parameters in π. It also derives the gradient for V, ϕ, f. Then, the employee sends the gradients for V, ϕ, π to the PPO gradient buffer and sends the gradients for f to curiosity gradient buffer (Line 20-21). The parameter optimization does not happen in employee threads. Instead, the employee waits for the notification from the chief and copies parameters from the global model (Line 22).

B. Pseudocode in Algorithm 2: Chief

In chief, for each update iteration k, it waits until all M employees have sent the gradient of θ, V and ϕ (Line 3), and sums up these gradients from different employees for updating global PPO model immediately (Line 4). It is same for the forward model f (Line 5-6). After, the chief clears up the global gradient buffers and notifies all M employees to start model parameter copying, as well as a new round of updating process (Line 8).

C. Pseudocode in Algorithm 3: Spatial Curiosity Model

The spatial curiosity model aims at generating the intrinsic reward based on the variety of state prediction. Specifically, it receives every worker's position $l_t^w = (x_t^w, y_t^w)$, $l_{t+1}^w = (x_{t+1}^w, y_{t+1}^w)$ and route planning action v_t^w. First, it obtains the embedding feature $\varphi(l_t^w)$ and $\varphi(l_{t+1}^w)$ (Line 2). After, it derives the prediction loss by Eqn. (16). Finally, it gets the intrinsic reward r_t^{int} as in Eqn. (17).

D. Testing Process

In a training process, the parameters in DNNs are periodically saved for testing. During testing, for all of the workers, we only use the trained policy network π to output their actions $\boldsymbol{a}_t$ simultaneously by policy model π, given the current state $\boldsymbol{s}_t$. Then, task environment gives the workers sparse extrinsic reward r_t^{ext}, and updates their state by $\boldsymbol{s}_{t+1}$. Therefore, our algorithm is a centralized control system which navigates multiple workers to collect data by a powerful VC server.

In detail, we use two kinds of neural networks, CNN and fully connected layers. The computation complexity of a L-layer CNN is $\mathrm{O}\big(\sum_{l=1}^{L} k_l^2 \cdot a_l^2 \cdot n_{l-1} \cdot n_l\big)$ where l is the index of a CNN layer, and k_l, a_l, n_{l-1}, n_l are kernel size, output feature map size, input channel and output channel respectively. For fully connected layers in actor-critic networks, the computation complexity is $\mathrm{O}\big(\sum_{l=1}^{L} n_{l-1} \cdot n_l\big)$ where n_l is the number of neural units in fully-connected layer l.

VII. Experiment

In this section, we first present the simulation setup, and then compared baselines, state-of-the-art approaches as well as metrics. Then, we show the comparison results, feature selection for curiosity model, and curiosity effect by visualization. All experiments are conducted on a Ubuntu 18.04 server with one NVIDIA RTX graphic card and Intel(R) Xeon(R) Gold 6238 CPU @2.10GHz.

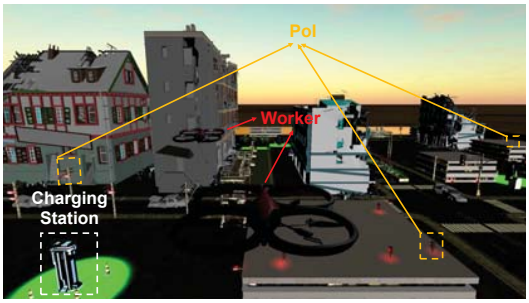
A. Setup

We consider the drone-assisted post-earthquake rescue, and created a 3D simulation environment by Unity 2019.2.5f1 in Fig. 2(a). Corresponding 2D map layout (i.e., crowdsensing space) is shown in Fig. 2(b), with four basic elements: collapsed buildings (obstacles), charging stations, sensors (PoIs) and drones (workers). In particular, sensors are audio life detection instrument and infrared radiation camera. We also design a hard exploration subarea at the bottom right corner, which represents semi-destroyed collapsed buildings, where drones should make efforts to go into that area through a narrow passageway to collect data from the embraced sensors. We generate sensor positions through a mixture of Gaussian distributions and a random distribution. For each sensor p, we generate its initial data value randomly, i.e., $0 < \delta_0^p < 1$. For drones, their positions are also randomly initialized, and their initial energy budget is $b_0^w = 40$ units, and set their sensing range g^w as 0.8. We set data collection rate $\lambda = 0.2$ as in Eqn. (1). For energy consumption, we set $\alpha = 1.0$ and $\beta = 0.1$ as in Eqn. (3). A charging station's effective charging range is 0.8, which represents the pump pipe length. For our sparse reward setting, we set $\epsilon_1 = 5\%$ for data collection and $\epsilon_2 = 40\%$ for energy charging. We set $\eta = 0.3$ for our curiosity model. Fig. 2(c) shows the illustrative moving trajectories for two drones.

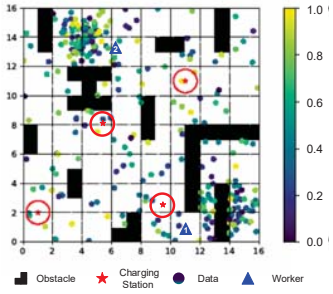
B. Compared Baselines and State-of-the-Art Approaches

- **D&C**: Lian *et al.* in [7] proposed a prediction-based algorithm to solve the maximization of task quality, called divide-and-concur (D&C). It includes next step condition into current time task assignment decision. In our case,

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.



(a) Scenario simulation for post-earthquake rescue.



(b) Considered 2D crowdsensing space.

(c) Attained trajectories for 2 drones and 4 charging stations.

Fig. 2: Experimental setup, simulation and trajectory visualization.

we first derive all the possible positions for workers at time slot $t+1$ and $t+2$, and calculate the expected collected data. After, we choose the actions that can maximize the expected collected data for time t.

- **Edics**: Our previous work in [20] proposed a multi-agent DRL-based algorithm to maximize the total collected data. We implement it by using W agents, each of which makes task assignment decision for one worker.
- **DPPO**: Heess *et al.* in [21] proposed a distributed PPO model (called DPPO) to train several simulated bodies on a diverse of challenging terrains to learn running, jumping, etc. This architecture facilitates the training process, and achieves a much better result than PPO itself. We implement DPPO and adopt the reward function in Eqn. (20). We also adopt the per-batch normalization of advantages mentioned in paper and set the number of employees as 8, batch size as 250.
- **Greedy approach**: For each time slot t, the server first derives all the possible positions for worker w at time $t+1$, and then calculates the corresponding collected data. After, the worker w travels to the specific position that maximizes the collected data while satisfying its current energy budget.

Note that both Edics and DPPO depend on the extrinsic and specific reward function definition, including the overall collected data, energy consumption and the geographical fairness. Therefore, We formulate their dense reward function as:

$$r_t^w = \frac{1}{W}\sum_{w=1}^{W}\left(\frac{q_t^w}{e_t^w} + \frac{\sigma_t^w}{b_0^w} - \tau_t^w\right), \tag{20}$$

where the first term indicates the reward for data collection, and the second term indicates the reward for charging energy, and the third term indicates the penalty for bumping into the obstacle or going beyond the space, respectively.

As discussed in Section III-B, we reuse these three evaluation metrics to evaluate the effectiveness of algorithms in our system, as: average data collection ratio κ, average remaining data ratio ξ, and energy efficiency ρ.

C. Impact of DNN Hyperparameters

We evaluate the impact of two most important hyperparameters in our model, namely (a) number of employees (in distributed computational architecture) and (b) updating batch size. As shown in Table II, the performance improves with more employees. However, from the training time in Fig. 3, we see that for 16 employees when batch size is 250, the running time is 45.5% significantly longer than that of 8 employees, but only 1.7% increases in ρ.Therefore, we choose the batch size of 250 and 8 employees.

TABLE II: Impact of two hyperparameters.

# of employees		1	2	4	8	16
batch size 50	κ	0.372	0.570	0.871	0.840	0.86
	ξ	0.6414	0.486	0.176	0.208	0.185
	ρ	0.100	0.189	0.402	0.380	0.395
batch size 125	κ	0.522	0.753	0.851	0.911	0.921
	ξ	0.5203	0.2886	0.19	0.1235	0.11
	ρ	0.176	0.385	0.397	0.445	0.445
batch size 250	κ	0.510	0.765	0.886	**0.927**	0.937
	ξ	0.5237	0.291	0.136	**0.117**	0.1
	ρ	0.173	0.376	0.440	**0.452**	0.460
batch size 500	κ	0.363	0.650	0.834	0.900	0.922
	ξ	0.6484	0.373	0.187	0.125	0.116
	ρ	0.139	0.323	0.393	0.435	0.448

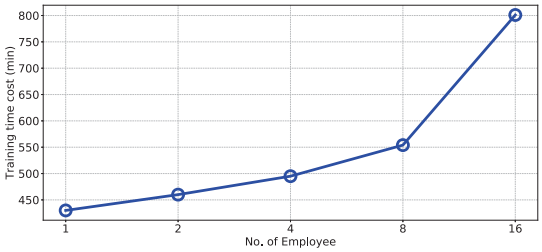
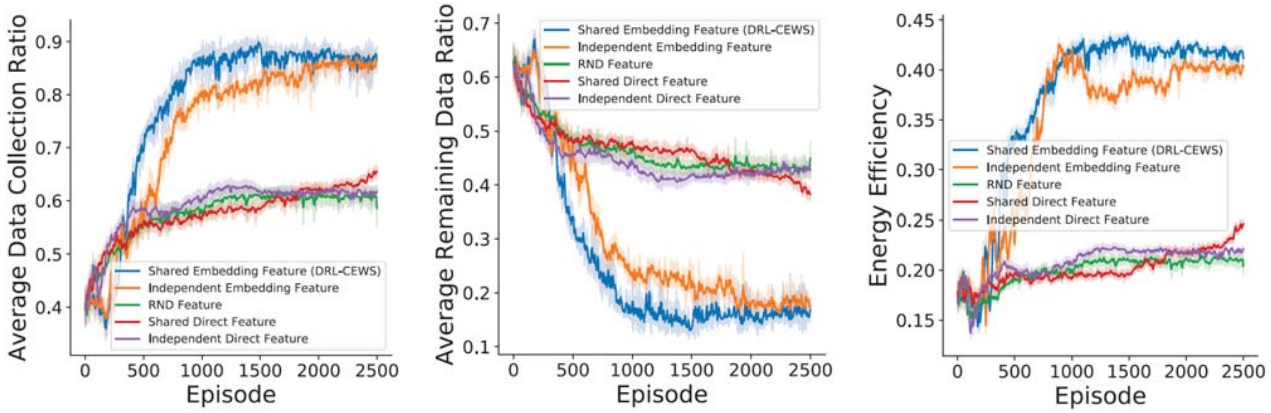


Fig. 3: Training time with different no. of employees.

D. Feature Selection for Curiosity Model

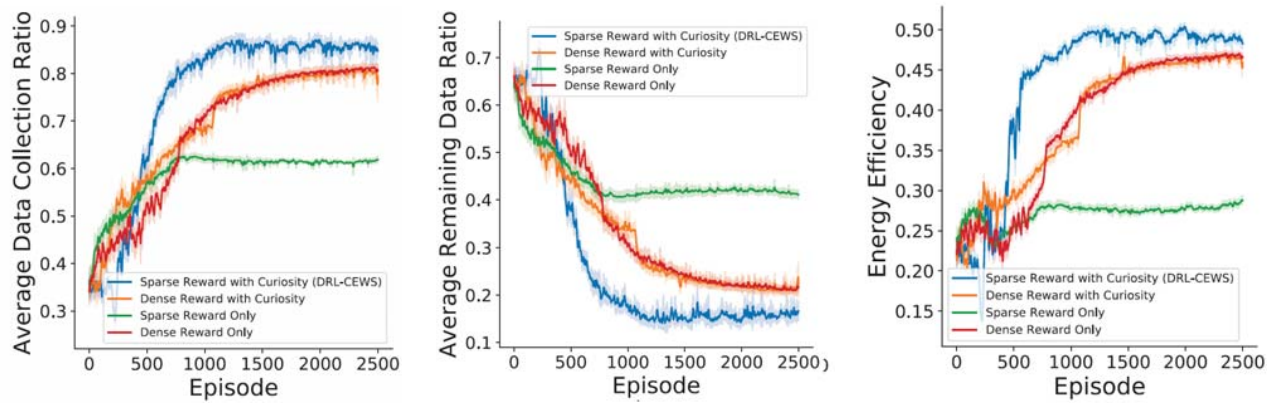
Burda *et al.* in [22] carried out large scale experiments to test the effectiveness of different representation networks for their curiosity model. Surprisingly, they found that a static randomly initialized CNN has a stable performance, and sometimes it does better than learned features. To this end, in our scenario, we consider two stable representations of a worker's spatial information, i.e., the direct feature and the embedding feature. The direct feature extractor directly scales a worker's position into range $(0, 1)$. The embedding features utilizes a static embedding layer, to map the position to a 8 dimensional spatial vector.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.



(a) κ (b) ξ (c) ρ

Fig. 4: Impact of feature selection for curiosity model.



(a) κ (b) ξ (c) ρ

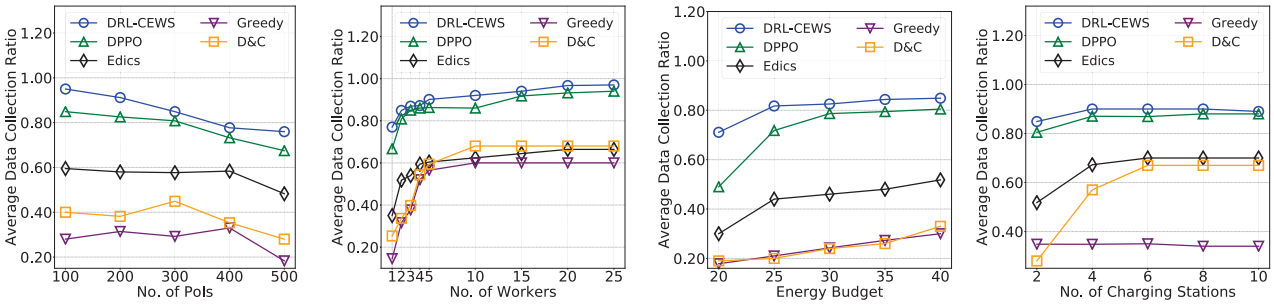
Fig. 5: Impact of dense and sparse reward w/ and w/o curiosity.

We also consider the curiosity model structure as an influencing factor. Specifically, we design two kinds of structures, called independent structure and shared structure. For the former, we have W independent curiosity models for every worker. The model for worker w takes w's position information as input, and generates intrinsic reward only for w. By contrast, in a shared structure, there is only one curiosity model, and takes the position information from worker 1 to W as inputs sequentially, and also outputs the intrinsic reward for workers orderly. Then, we combine the above feature extractors and model structures, and form four kinds of curiosity feature extractors, i.e., shared embedding feature, shared direct feature, independent embedding feature, and independent direct feature. Besides, we use a state-of-the-art curiosity model called random network distillation (RND) [23], which models the next state prediction error as the intrinsic reward. We set $W = 2$ and $P = 200$ in our simulation.

Results are shown in Fig. 4. It is obvious that an embedding feature is much better than the direct representations. For example, in Fig. 4(a), κ at episode 2,500 for the embedding feature is 27% and 25% higher than the direct features in shared and direct structure, respectively. This is because that two locations could be far away from each other in the embedding space, even if these two points are close physically. Also, this can bring a larger intrinsic reward, which facilitates the training process to converge to an optimal policy. Furthermore, we see that RND feature is inefficient in our system. One of the possible explanation is there are multiple workers in our system, and the size of the state is too complex for modeling. It also indicates that modeling the conjoint features of multiple workers in curiosity model may be inefficient.

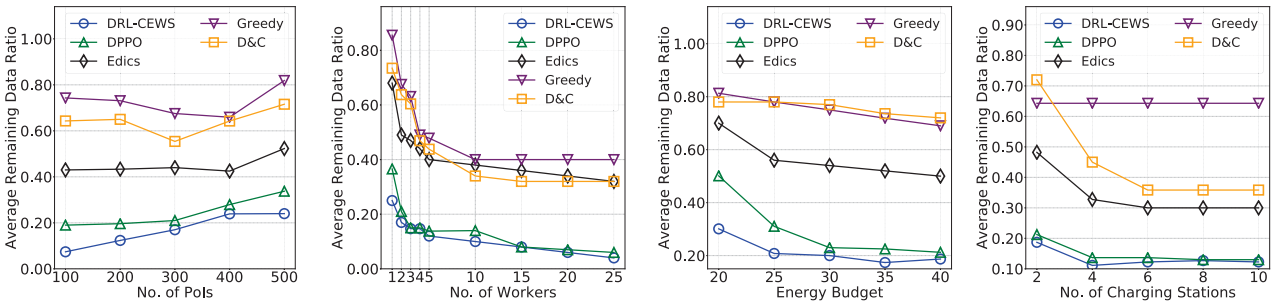
Besides, we also notice that the shared model structure converges more quickly than independent structure and achieve a better performance. This is because that the former allows different workers to share their historical information by using common parameters, and a worker could discover a better policy by following others' experiences. Also, the number of parameters in shared model does not grow with more workers. On the other hand, the space complexity for independent structure will be multiplied with more workers. Hence, we decide to select the shared embedding feature for our curiosity model considering its effectiveness and fixed space complexity.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.



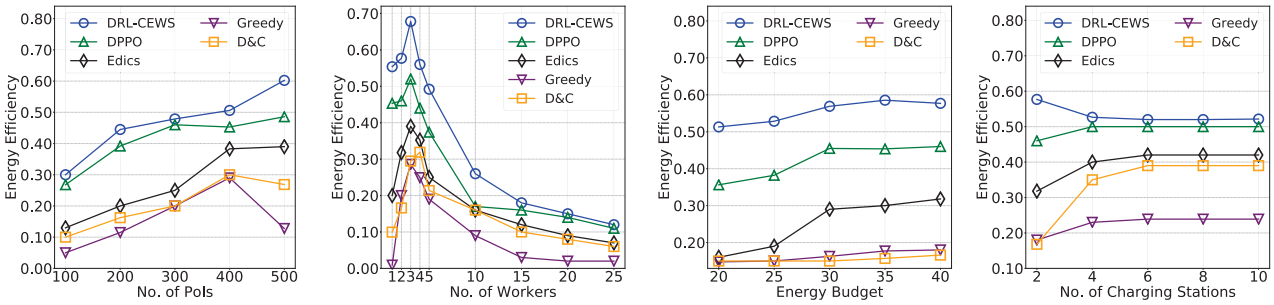
(a) Varying number of PoIs (b) Varying number of workers (c) Varying energy budget (d) Varying no. of charging stations

Fig. 6: Results of average data collection ratio κ.



(a) Varying number of PoIs (b) Varying number of workers (c) Varying energy budget (d) Varying no. of charging stations

Fig. 7: Results of average remaining data ratio ξ.



(a) Varying number of PoIs (b) Varying number of workers (c) Varying energy budget (d) Varying no. of charging stations

Fig. 8: Results of energy efficiency ρ.

E. Impact of Different Reward Mechanisms with Curiosity

We set $W = 2$ and $P = 300$ in our simulation and evaluate our curiosity model in dense and sparse extrinsic reward mechanisms, respectively, as shown in Fig. 5. First, we observe that all four methods converge to a stable performance after 2,000 episodes. The combination of sparse reward and curiosity acts the best over κ, ξ and ρ. For example, in Fig. 5(c), "sparse reward with curiosity" achieves $\rho = 0.48$, which is 4.35%, 77.8% higher than that of "dense reward only" (whose $\rho = 0.46$), and "sparse reward only" (whose $\rho = 0.27$). The result of sparse reward only indicates that DRL-based method does not work well in a sparse reward environment. However, with the help of intrinsic reward design, DRL-CEWS converges quickly (i.e., the sparse reward with curiosity achieves a stable performance after 1,500 episodes) and is more effective.

We also notice that the curiosity does not bring a large improvement in the dense reward environment, except that it is rising faster in the initial stage of training (before 1,000 episodes). They achieve the almost the same performance, as $\kappa = 0.81$, $\xi = 0.79$, $\rho = 0.46$ in episode 2,500. Hence, the inclusion of curiosity indeed can facilities the training process and speed up the convergence in dense reward environment. However, curiosity has limited effect in dense reward environment, where the well-designed reward function plays the main role.

F. Impact of Number of PoIs

We compare our solution with other four algorithms in Fig. 6(a), Fig. 7(a) and Fig. 8(a). We fix $W = 2$, and vary P from 100 to 500 without changing the distribution of PoIs. We observe that the DRL-CEWS achieves the best performance in κ, ρ and ξ. For example, when $P = 500$, it achieves an

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.

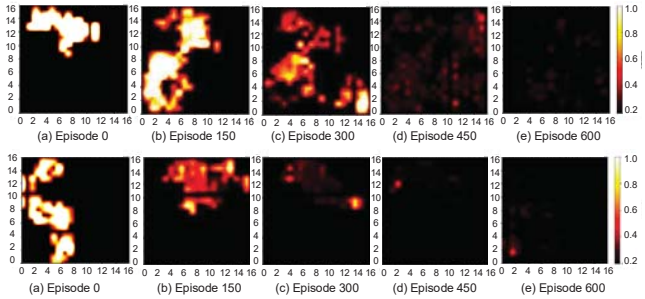


Fig. 9: Curiosity visualization by showing the curiosity value for a worker at its passed location, where the top five figures show the result of DRL-CEWS in episode [0,150,300,450,600], and the bottom five figures show that of DPPO solution.

energy efficiency 0.60, which is 23.98%, 55.83%, 123.07%, 371.28% higher than DPPO, Edics, D&C and Greedy methods, respectively. More PoIs impose more challenges for workers, because rich data area may lead a worker to a local suboptimal solution with bigger extrinsic reward. However, this may lead to a lower κ and ρ in a long term. Greedy and D&C fail to achieve the trade-off between data collection and energy charging, because they lack the ability to predict the multiple steps ahead. However, in our solution, curiosity model helps to drive the workers explore sparsely distributed subareas by generating the intrinsic reward, for the long run.

As shown in Fig. 6(a), the average data collection ratio decreases with more PoIs. For example, when $P = 100$, κ is 25% more than that of when $P = 500$ for our method. This is because the worker's data collection capability is limited by its energy budget, and thus the same number of workers may not be sufficient to collect all data with more PoIs. We also notice that DRL-CEWS achieves a lower average remaining data ratio, compared with the other four algorithms in Fig. 7(a). For example, when $P = 100$, $\xi = 0.07$ for DRL-CWES, which is 0.36 and 0.67 lower than that of Edics (whose $\xi = 0.43$) and Greedy (whose $\xi = 0.74$), respectively.

G. Impact of Number of Workers

We vary W from 1 to 25, and fix $P = 300$ in Fig. 6(b), Fig. 7(b) and Fig. 8(b). Here a worker refers to drones or driverless cars, that have stronger capability for sensing which usually can service a group of human participants simultaneously, as a mobile base station. From Fig. 6(b), we can see that the average data collection ratio increases when more workers are deployed. Specifically, for Edics, $\kappa = 0.60$ with 5 workers, which is 71.42% higher than $\kappa = 0.35$ when $W = 1$. This is because with more workers, they learn to be responsible for a subarea, which improves the overall collected data quantity in a given time period. Furthermore, our method achieves the highest energy efficiency in Fig. 8(b). This is collectively achieved by our distributed Chief-Employee computational architecture for generating more effective experiences and updating DNN parameters, and spatial curiosity model with sparse reward design for better long-term exploration.

We also find that ρ decreases when $W > 4$. For example, when $W = 25$, the energy efficiency for DRL-CEWS is 0.12, which is 0.38 lower than $\rho = 0.49$ when $W = 5$. This is because, after most data are collected, workers still should move around to search for the corner cases. As a result, with more than enough workers, the energy efficiency decreases.

H. Impact of Energy Budget

We explore the impact of energy budget. Results are shown in Fig. 6(c), Fig. 7(c) and Fig. 8(c). We see that DRL-CEWS remains high performance even with little energy budget. For example, when energy budget is 20, $\kappa = 0.71$ for DRL-CEWS, which is 22%, 41%, 48%, 53% higher than that of DPPO (whose κ=0.49), Edics (whose κ=0.30), D&C (whose κ=0.23) and Greedy (whose $\kappa = 0.18$). This is because even with little energy budget, DRL-CEWS can still find a good trade-off between energy charging and data collection with the help of spatial curiosity model.

I. Impact of Number of Charging Stations

We vary different number of of charging stations. Results are shown in Fig. 6(d), Fig. 7(d) and Fig.8(d). From Fig. 6(d), κ increases from 2 to 6 stations, and remains unchanged from 6 to 10. For D&C, $\kappa = 0.67$ with 6 charging stations, which is 19% higher than $\kappa = 0.28$ with 2 charging stations, and remains unchanged with 6 to 10 charging stations (where $\kappa = 0.67$). For DRL-CEWS and DPPO, they all find an energy efficient route with necessary charging stations. Therefore, excessive charging stations have little influence on them. For Edics and D&C, more charging stations help explore larger regions, than that of DRL-CEWS and DPPO. We also notice that charging station amount has nearly no influence on Greedy approach. This is because with Greedy, workers are easily trapped in a small region; thus they may use up their energy budget and fail to find other charging stations.

J. Curiosity Visualization for DRL-CEWS and DPPO

In order to show how our curiosity model helps to obtain a good task assignment policy, we visualize the curiosity value for each location where a worker has visited in the past as a heat map, i.e., $W = 1$, when $P = 300$, as in Fig. 9.

For both methods, the curiosity value r_t^{int} is decreasing when the the training process progresses, as shown the brightness is weakened. This also means that the policy is becoming more stabilized, since at each specific location, the worker's action is almost fixed. As a result, the predicted loss $Loss^f$ is minimized. From these figures, we can explicitly show how curiosity helps to derive an optimal policy. For example, it is very bright in episode 300, since the worker has never been to that subarea before, and thus the loss for the forward model is large, which brings a high intrinsic reward. Referring to the entrance of the corner area in Fig. 2(b), our DRL-CEWS gives the worker a large intrinsic reward when it visited the corner. From the contrasted heat map between our method and DPPO, we can clearly see the benefit of proposed spatial curiosity model that encourages the policy exploration. Thus the area

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.

of the brightness space is larger than that of DPPO. This is further confirmed by illustrative trajectories in Fig. 2(c).

VIII. Conclusion

In this paper, we explicitly consider the VC problem with multiple unmanned workers to perform a sensing task, with the presence of multiple charging stations. We proposed a DRL-based approach, called DRL-CEWS, which utilizes a distributed chief-employee computational architecture with PPO agent to derive the optimal policy. We also include a novel spatial curiosity model with sparse reward design for better exploration so that an optimal policy in large crowdsensing space with unevenly distributed data can be obtained. We conducted extensive experiments to show that DRL-CEWS outperforms all other baselines and state-of-the-art methods, and we showed the impact of two hyperparameters and process of feature selection. Finally, we visualized the benefits of our curiosity model.

Acknowledgement

This paper was supported by National Natural Science Foundation of China (No. 61772072).

References

[1] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: a survey," *The VLDB Journal*, pp. 1–34.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[3] Z. Liu, L. Chen, and Y. Tong, "Realtime traffic speed estimation with sparse crowdsourced data," in *IEEE ICDE'18*, 2018, pp. 329–340.

[4] H. Hu, G. Li, Z. Bao, Y. Cui, and J. Feng, "Crowdsourcing-based real-time urban traffic speed estimation: From trends to speeds," in *IEEE ICDE'16*, 2016, pp. 883–894.

[5] Z. Chen, P. Cheng, Y. Zeng, and L. Chen, "Minimizing maximum delay of task assignment in spatial crowdsourcing," in *IEEE ICDE'19*, 2019, pp. 1454–1465.

[6] H. Hu, Y. Zheng, Z. Bao, G. Li, J. Feng, and R. Cheng, "Crowdsourced poi labelling: Location-aware result inference and task assignment," in *IEEE ICDE'16*, 2016, pp. 61–72.

[7] X. Lian, L. Chen, C. Shahabi, and P. Cheng, "Prediction-based task assignment in spatial crowdsourcing," in *IEEE ICDE'17*, 2017, pp. 997–1008.

[8] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv, "Adaptive dynamic bipartite graph matching: A reinforcement learning approach," in *IEEE ICDE'19*, 2019, pp. 1478–1489.

[9] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," *IEEE ICDE'16*, pp. 49–60, 2016.

[10] C. H. Liu, Z. Dai, Y. Zhao, J. Crowcroft, D. O. Wu, and K. Leung, "Distributed and energy-efficient mobile crowdsensing with charging stations by deep reinforcement learning," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2019.

[11] Z. Zhou, J. Feng, G. Bo, A. Bo, and M. Guizani, "When mobile crowd sensing meets uav: Energy-efficient task assignment and route planning," *IEEE Transactions on Communications*, vol. 66, no. 11, pp. 5526–5538, 2018.

[12] K. Liu, J. Peng, L. He, J. Pan, S. Li, M. Ling, and Z. Huang, "An active mobile charging and data collection scheme for clustered sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5100–5113, 2019.

[13] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 9, pp. 2296–2319, 2016.

[14] C. Wang, X. Gaimu, C. Li, H. Zou, and W. Wang, "Smart mobile crowdsensing with urban vehicles: A deep reinforcement learning perspective," *IEEE Access*, vol. 7, pp. 37 334–37 341, 2019.

[15] S. F. Abedin, M. G. R. Alam, A. K. Bairagi, A. Talukder, and C. S. Hong, "Uav-assisted intelligent crowdsourcing in natural calamity," in *The Int'l Symp. on Perception, Action, and Cognitive Systems*, 2016.

[16] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 1984.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArxiV*, vol. abs/1707.06347, 2017.

[18] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," *arXiv preprint arXiv:1802.01561*, 2018.

[19] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *IEEE CVPR'17*, 2017, pp. 16–17.

[20] C. H. Liu, Z. Chen, and Y. Zhan, "Energy-efficient distributed mobile crowd sensing: A deep learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1262–1276, 2019.

[21] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, M. Riedmiller *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv*, vol. abs/1707.02286, 2017.

[22] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," in *IEEE ICLR'19*, 2019.

[23] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," in *IEEE ICLR'19*, 2019.

Authorized licensed use limited to: BEIJING INSTITUTE OF TECHNOLOGY. Downloaded on October 10,2024 at 02:16:32 UTC from IEEE Xplore. Restrictions apply.