IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Smart Mobile Crowdsensing with Urban Vehicles: A Deep Reinforcement Learning Perspective

CHAOWEI WANG, Member IEEE, XIGA GAIMU, CHENSHENG LI, HE ZOU, WEIDONG WANG, Member IEEE

School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Chaowei Wang (e-mail: wangchaowei@bupt.edu.cn).

This work was supported by the National Key Research and Development Program of China (2017YFC0804404).

ABSTRACT Mobile crowdsensing (MCS) is a promising sensing paradigm based on mobile node which provides solution with cost-effectiveness to perform urban data collection. To monitor the urban environment and facilitate the municipal administration, more and more applications adopt vehicles as participants to carry out MCS tasks. The performance of the applications highly depends on the sensing data which is influenced by the recruiting strategy on vehicles. In this paper, we propose a novel vehicle selection algorithm to maximize the sensing range with limited cost while the vehicle selection problem was proved to be NP-complete. Specifically, we modeled the interaction between MCS server and candidate vehicles as a Markov Decision Process (MDP) and formulated the maximum spatial temporal coverage (STC) optimization as a deep reinforcement learning process. The performance of our deep reinforcement learning based vehicle selection (i.e. DRLVS) algorithm is evaluated with real trajectory dataset. The numerical result indicates that the proposed algorithm achieves an optimal solution and maximizes the STC.

INDEX TERMS Mobile crowdsensing, spatial-temporal coverage, deep reinforcement learning

I. INTRODUCTION

With the development of Internet of things (IoT) infrastructure and massive growth of mobile sensing devices, such as smart phones/vehicles, various applications, that support smart city components based on big data were created [1], [2]. As a typical part of smart city, vehicular network has become the platform for those applications. To meet the increasing demand of data in IoT applications, mobile crowd sensing (MCS) has become a promising paradigm for large scale sensing in urban areas [3]. Specifically, due to the mobility and the scope of activity, vehicles have been adopted as the participants of the MCS applications [20].

By using the sensing ability and the mobility of the vehicles, it is easy to collect location based sensing data without deploying fixed sensor nodes [5]. In particular, a vehicle based MCS system consists of a MCS server as a recruiter and the vehicles as the participants. The recruiter recruits participants to monitor the surrounding environment and contribute to mobile crowdsensing tasks. Meanwhile, the participants are rewarded according to their contribution and the incentive mechanisms of the MCS system [6], [7]. MCS was widely studied from different perspectives in recent years. In [8], researchers studied crowdsensing applications and classify them into three categories: environmental, infrastructure and social science applications. In environmental applications, participants collect the information of natural phenomena like pollution, air condition and water level monitoring [9]. Infrastructure applications involve large scale monitoring related to public infrastructure including observing traffic congestion [10], [11]. In social science applications, MCS nodes share sensed information among themselves. Example includes sharing road condition with other nodes for choosing best routine [12]. On the other hand, the performance of MCS application highly relies on the reliability of MCS system, as some participants may launch fake sensing attacks which lead to security and privacy concern [13], [14]. In [16], a comprehensive security and privacy protecting MCS architecture was proposed to guarantee the deployment of MCS applications. Despite the critical reliability problem of MCS, the key problem is selecting participants and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905263, IEEE Access

collecting large scale of data. In [17], Zhao et el. designed two online incentive mechanisms to attract more participants with budget constraint. An active participant recruitment method was presented to improve coverage area and effectiveness of service in [7]. In [18] Peng et al. proposed a data quality based incentive mechanisms, which estimate the quality of sensing data, and participants are rewarded based on their effective contribution.

In [21], Zhu et al. studied the base station deployment problem for maximizing the expected coverage by the vehicular network. In [4], He et al. presented a participant recruitment strategy for vehicle based crowdsensing, considering both current location and future estimated trajectories of the vehicles, and they consider the spatial coverage and temporal coverage respectively. A combination of spatial and temporal coverage was studied in [6] based on public transport with predictable trajectory. In [22], Liu et al. proposed a vehicle selection algorithm which can collect comprehensive spatial temporal sensing data.

Generally, the methods to maximize the sensing data or sensing coverage both belong to NP-complete problem which is proved in [4], [6], [21] and [22], and choosing participants is always modeled as an optimization problem. Nevertheless, to find the optimal or sub-optimal solution in effective time, the performance of the proposed algorithms has to be trade off in the above literatures. Due to the ability of learning optimal policy from the interaction between agent and the environment, deep learning (DL) and reinforcement learning (RL) have become popular methods to solve optimization problem. In [7], Zhou et al. used DL to validate data from the original sensed dataset and improve the efficiency and robustness of the MCS system. In [15], Xiao et al. investigated secure MCS and proposed a deep learning (DL) based method to improve approaches to MCS security including authentication, intrusion detection and privacy protection. In [25], Chen et al. proposed a multi-agent reinforcement learning based scheme to derive an online sensing policy to maximize the payoffs of the participants.

In this article, considering the feature of the vehicle based MCS problem, we proposed a deep reinforcement learning based vehicle selection scheme (DRLVS) to obtain an optimal solution and maximize the spatial temporal coverage. Specifically, we first modeled interactions between MCS server and sensing environment as a Markov Decision Process, and formulated the vehicle selection problem with constraint to an DRL problem. Then with the objective of maximizing the spatial temporal coverage, we used DRL to obtain optimal policy and maximize the STC. The experimental results show that our proposed algorithms can get an optimal solution and maximize the STC.

The rest of this paper is organized as follows. Section II describes the system model of MCS system. In Section III, the details of the proposed DRL-based vehicle selection scheme are illustrated. Simulation results are presented and discussed in Section IV. Finally, Section V concludes the paper.

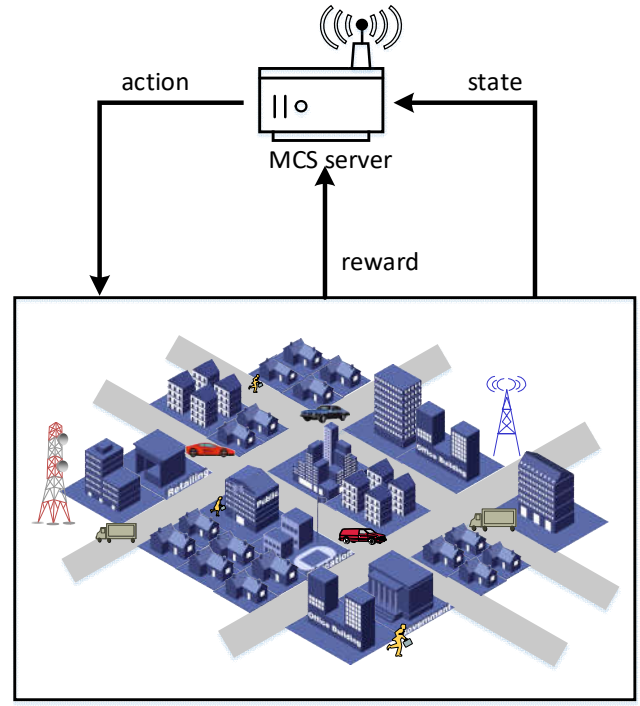

FIGURE 1. MDP model of mobile crowdsensing system.

II. SYSTEM MODEL

Our vehicle mobile crowd sensing system is shown in Fig.1, which consists of a MCS server and vehicles equipped with multiple sensors. The MCS server broadcasts sensing tasks and selects several vehicles as participants according to the choosing policy. Specifically, the participants aggregate the sensing data to MCS through access points.

As shown in Fig.2, we divide the geographic area into lattice cells which are called regions of interests (ROIs). Let R denotes the set of ROIs, $R=\{r_1,r_2,\cdots,r_n\}$ and each of ROIs denotes a sensing unit, so there are n sensing units corresponding with n ROIs in total. Suppose that there are k vehicles in vehicle set $V=\{v_1,v_2,\cdots,v_k\}$, we can draw k full lines as their trajectories traveling through the different ROIs. In our model, the MCS server collects the information sensed by vehicles every Δt, and we can get discrete sensing period set $T=\{t_1,t_2,\cdots,t_m\}$, in which the MCS server collects information. The dots on the trajectory indicates the position of the vehicles at sensing period t_i. Specifically, despite the vehicle's position is at the center or the edge of the ROI, we assume that a vehicle can sense all the information of a ROI during the sensing period. Therefore, with V and T, we can express the trajectories of vehicles as a matrix:

$$L(V)=\begin{matrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{matrix}\begin{bmatrix} l_1(t_1) & l_1(t_2) & \cdots & l_1(t_m) \\ l_2(t_1) & l_2(t_2) & \cdots & l_2(t_m) \\ \vdots & \vdots & \vdots & \vdots \\ l_k(t_1) & l_k(t_2) & \cdots & l_k(t_m) \end{bmatrix}, \quad (1)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905263, IEEE Access

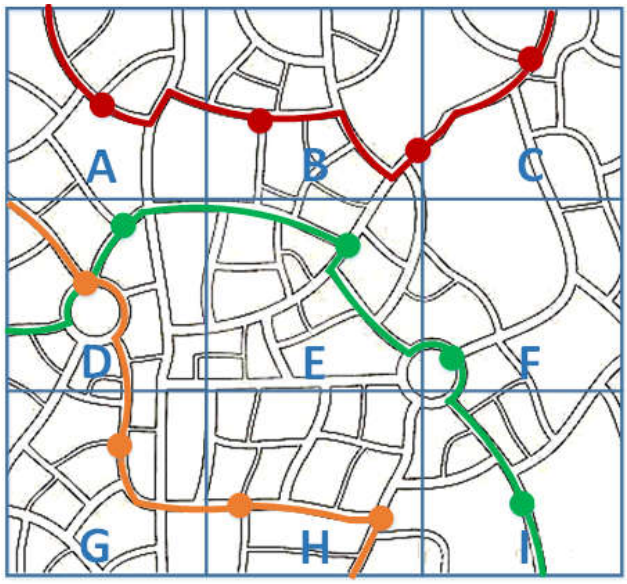

FIGURE 2. **Demonstration of region of interests and trajectories.**

where $l_i(t_j) \in R$. Each row vector indicates the trajectory of a certain vehicle at sensing period set T, and each column vector denotes the locations of all vehicles at a certain period.

In practice, MCS system is always involved in incentive mechanisms, and the key problem is allocating the rewards to the participants with a specific rule.

Definition 1: Sensing Reward (SR) In a MCS system, the recruiter rewards the participants for performing the sensing tasks. In our model, the MCS server rewards the vehicles which are selected to participate. As to the amounts of rewards, it is equal to the real cost generated by participants during the sensing period [7]. The costs of vehicles are different, and they can be acquired by online bidding. Let C denotes the reward vector:

$$C = \{c_1, c_2, \cdots, c_k\}, \tag{2}$$

and each of them is related to a candidate vehicle.

We define an indication vector Φ to indicate whether a vehicle is selected or not,

$$\Phi_i = \begin{cases} 1, & v_i \in \Omega \\ 0, & other \end{cases}, \ i \in \{1, 2 \cdots n\}, \tag{3}$$

where $\Omega \subseteq V$ denotes the set of selected vehicles. Thus, the total reward to selected vehicles is calculated as

$$\omega(\Omega) = \Phi \cdot C^T. \tag{4}$$

For practical consideration, there is a total budget while recruiting the vehicles for sensing task. The MCS server decides the total sensing reward i.e. the budget of the sensing tasks $C_{\max}$. The selected vehicles should satisfy $C(\Omega) \le C_{\max}$.

Definition 2: Spatial Temporal Coverage (STC) STC strongly affects the crowdsensing quality. As we have mentioned above, a certain vehicle is able to cover a ROI where it is located. Thus, according to the trajectory matrix, the total STC is the total number of different regions that covered during all the sensing periods,

$$STC(\Omega) = \sum_{t_j \in T} \bigcup_{v_i \in \Omega} l_i(t_j). \tag{5}$$

To explain the equation above, we give an example according to Fig.2. As shown in the figure, there are three vehicles $\{v_1, v_2, v_3\}$ traveling in the sensing area during the sensing period, and the trajectories are $\{A, B, B, C\}$, $\{D, E, F, I\}$, $\{D, G, H, H\}$ respectively. Then we can get the trajectory matrix as according to (1):

$$L(v_1, v_2, v_3) = \begin{bmatrix} A & B & B & C \\ D & E & F & I \\ D & G & H & H \end{bmatrix}. \tag{6}$$

At the first sensing period, v_2 and v_3 are in same ROI D, so according to (6), the duplicated ROI would be counted as 1. Assume that we recruit $\{v_1, v_2, v_3\}$, the STC is $STC(v_1, v_2, v_3) = 11$.

With the system model, we can formulate the vehicle selection problem as an optimization problem to maximize the STC with limited budget.

Definition 3: Vehicle Selection Problem (VSP) Vehicle selection problem is to select a set of vehicles from given candidate participants to maximize the STC and subject to the budget constraint $C_{\max}$.

Given:

k candidate vehicles, $V = \{v_1, v_2 \cdots v_k\}$ with the corresponding cost $C = \{c_1, c_2 \cdots c_k\}$.

n ROIs, $R = \{r_1, r_2 \cdots r_n\}$.

m sensing periods, $T = \{t_1, t_2 \cdots t_m\}$.

The trajectory of candidate vehicles $L(V)$.

Objective:

Select a set of vehicles $\Omega \subseteq V$ to achieve $\max STC(\Omega)$.

Subject to: $C(\Omega) \le C_{\max}$.

III. DEEP REINFORCEMENT LEARNING BAESED VEHICLE SELECTION SCHEME

In this section, based on above discussions, we propose a DRL based vehicle selection algorithm to maximize the STC in MCS system. First, we give a brief introduction about DRL, then we model the vehicle selection problem into MDP. Finally, we show the implementation of the proposed scheme.

A. DEEP REINFORCEMENT LEARNING

Reinforcement learning is a part of machine learning focusing on getting an optimal policy π to solve certain tasks. The agent (i.e. human, MCS server) observes the environment and executes an action, then the environment transfers into next state with a reward given back to the agent. According to the reward, the agent adjusts the action for the next iteration.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905263, IEEE Access

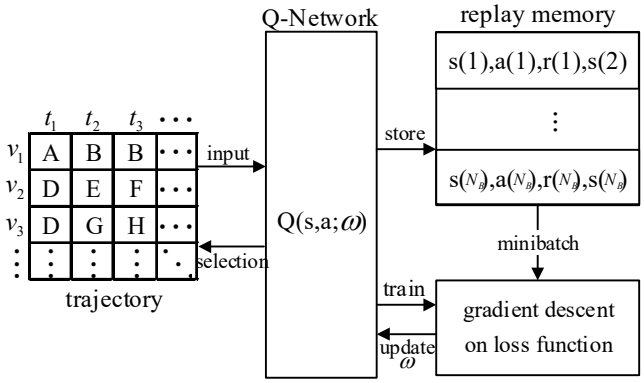

FIGURE 3. **The flowchart of DRLVS algorithm.**

The objective of reinforcement learning is to find an optimal state-action policy to maximize the cumulative future rewards and transfer the environment into optimal state [23]. Specifically, from the cumulative future rewards, we can calculate the value of an action which is expressed as action value function (Q-function):

$$Q^{\pi}(s,a)=\mathbb{E}\left[r_{t+1}+\lambda r_{t+2}+\lambda^2 r_{t+3}+\cdots\middle|s_t=s,a_t=a,\pi\right], \tag{7}$$

where a_t and s_t denotes action and state at time t, and r_t is the immediate reward after taking action a_t under state s_t. The Q-function denotes the expectation of the sum of r_t discounted by λ each time t under policy π. With these definitions, the optimal policy should satisfy:

$$Q^{*}(s,a)=\max_{\pi}Q^{\pi}(s,a). \tag{8}$$

By selecting the action with maximum Q-value in each iteration, the agent learns to perform a high Q-value action and come up with an optimal policy.

It should be noted that Q-value of all state-action pair is updated recursively. In practice, we use Q-learning to update Q-value with learning rate α and it is denoted as:

$$Q_{\tau+1}(s,a)=Q_{\tau}(s,a)+\alpha(r+\lambda\max_{\pi:s\to a}Q_{\tau}(s',a')-Q_{\tau}(s,a)). \tag{9}$$

Above mentioned RL method like Q-learning updating rule is an extension of Bellman equation [24], which is used to deal with the problem of small state/action space. However, in our vehicle selection scenario, the state/action space is too large to use Q-learning. Therefore, we introduce deep Q-network (DQN) [23], a typical instance of DRL, which combines Q-learning with deep learning to solve our problem. DQN uses deep neuro networks (DNN) to approximate action-value function:

$$Q(s,a)\approx Q(s,a;\omega) \tag{10}$$

in which ω denotes the parameter weights of DNN. The neuro networks are trained by minimizing the loss function in each iteration to get the parameter ω:

$$\omega:\min L(\omega)=\mathbb{E}[(r+\lambda\max_{a'}Q_{\tau}(s',a';\omega))-Q(s,a;\omega)]. \tag{11}$$

It should be noted that the DRL can only solve the problem that can be formulated as a Markov Decision Process (MDP). In the next subsection, we give the details of MDP for vehicle selection problem.

B. MDP MODEL OF VSP

Generally, the problem to be solved in RL can usually be regarded as a MDP, in which the next state is only decided by the current state and the action implemented on it. In this paper, we model the interaction between MCS server and the environment as Fig.1. Next, we illustrate it in three aspects: state, action and reward.

1) SYSTEM STATE

In this paper, MCS sever acts as an agent, who observes the environment state i.e. the trajectories of candidate vehicles, and take action to select the vehicles set for maximizing the STC.

As shown in system model, STC can be calculated from the trajectory matrix, which is an unprocessed form of the state. In our formulation, input of the DNN i.e. the processed form of state is one-dimensional vector that indicates the covered times of each ROI. For example, for trajectory in (6), the processed form of state is:

$$s=\left[N_A,N_B,N_C,\cdots,N_I\right]_{1\times 9}, \tag{12}$$

where N denotes the covered times of an exact ROI (like $A,B,\cdots,I$). In particular, the state of the (6) is:

$$s=[1,2,1,1,1,1,1,2,1]. \tag{13}$$

Obviously, the STC can be expressed as:

$$STC=\sum_{i\in n}N_i. \tag{14}$$

In general, the state space is 2^k-1, and k is the number of vehicles.

2) ACTION

As mentioned above, the action in our model is the selection of vehicles. The MCS sever executes an action according to the policy, then the environment responds to the action and presents new state to the agent. In practice, just like the unprocessed state, original action is a one-dimensional vector as well:

$$a=\left[q_1,q_2,\cdots,q_k\right],\ q_i\in\{0,1\},\ i=1,2,\cdots,k, \tag{15}$$

where q_i indicates whether the i-th vehicle is selected:

$$q_i=\begin{cases}1, & selected\\ 0, & not\ \ selected\end{cases}. \tag{16}$$

It is obvious the action space is also 2^k-1. However, we preprocessed the original action corresponding to the DNN input. First, we build an action matrix:

$$a_{matrix}=\begin{bmatrix}1&0&\cdots&0\\0&1&\cdots&0\\\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&1\end{bmatrix}. \tag{17}$$

The dimension of the action matrix is $\left[\left(2^k-1\right)\times\left(2^k-1\right)\right]$, and each row of it (i.e. a vector) is an action input for DNN. So there is a one-to-one match between the action and state.

3) REWARD

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905263, IEEE Access

In subsection A, we mentioned that when the agent selects an action, it gets a reward. Our optimization goal is to maximize the STC, and the reward is defined as:

$$r = \begin{cases} \phi^+, & STC(\Omega) \geq \eta \\ \phi^-, & else \end{cases}, \tag{18}$$

where η is the threshold of the STC, which is dynamically updated during the training process, and ϕ^+ is a positive reward value, while ϕ^- is a negative reward value. When the $STC(\Omega)$ is larger than η, the action would get a positive reward, and the weights of Q-network would be updated. After a number of iterations, the action with high reward is highlighted, and the probability of being chosen increases. When the training process ends, the action with highest reward is selected and the corresponding vehicle set is optimal for the VSP problem.

C. IMPLEMENTATION OF PROPOSED SCHEME

After defining the system model, state space, action space and reward, we illustrate the implementation of our DRL-based vehicle selection scheme.

Fig. 3 shows the flowchart of proposed DRLVS algorithm. We adopt DNN which has three hidden layers to approximate the non-linear function. There are 256, 256 and 512 neurons in each hidden layer respectively. In the first two hidden layers, the activation functions are rectified linear units (ReLUs). In the third hidden layer, the activation function is tanh function.

TABLE 1
THE PROPOSED MOBILE SENSING CROWDSENSING SCHEME

Algorithm I Implementation of DRL-based vehicle selection scheme

Initialize the capacity of replay memory to N_B
Initialize the parameters of Q-network with random weights
for training steps = 1, 100000 **do**
  Randomly choose a set of vehicles as beginning state s
  Randomly generate a probability p
  if $p \geq \varepsilon$
    Choose $a(t) = \arg\max_a Q(s, a; \omega)$
  else
    Randomly choose an action
  Execute action then obtain the reward $r(t)$ and next state $s(t+1)$
  Store $(s(t), a(t), r(t), s(t+1))$ in replay memory
  if $t \geq N_D$
    Randomly sample a minibatch of data from replay memory
    Obtain $Q_{target} = r + \lambda \max_{a'} Q_t(s', a'; \omega)$
    Perform a gradient descent step on loss function (11)
    Update the parameters of Q-network
  end if
end for
Output the maximum STC

To improve the performance of the proposed scheme, experience replay is used to train the Q-network. In this paper, we store the state, action and reward i.e. $(s(t), a(t), r(t), s(t+1))$ generated by the Q-network into replay memory, and we set the capacity of replay memory as N_B. When the number of stored state, action and reward set is larger than N_D, our algorithm randomly sample N_M (the size of minibatch) experience data from replay memory in each iteration and start training the Q-network.

In this paper, we use ε-greedy strategy to balance the exploration and exploitation. During the training process, the exploration rate decreases from an initial ε_s to an end ε_e linearly. The detail of proposed DRL-based vehicle selection algorithm is illustrated in the Table 1.

The above proposed algorithm selects an optimal vehicle set by training the Q-network to perform the mobile crowdsensing. Its STC performance comparing with existing algorithms is simulated in next section.

V. SIMULATION RESULTS

To evaluate the performance of our algorithm, we simulated the performance of proposed DRLVS algorithm in this section. First, the simulation parameters and settings of deep reinforcement learning is shown in Table 2. Then, the total STC is simulated respectively under the different settings of the number of vehicles, the sensing reward and sensing periods.

TABLE 2
THE SIMULATION PARAMEERS

Parameters of DQN	**Values**
Replay memory capacity N_B	5000
The maximum training steps	100000
Experience tuple size N_D	3000
Minibatch size N_M	32
Activation function of hidden layers	ReLUs
Activation function of output layers	tanh
The training optimizer	Gradient Descent
The learning rate	0.0001
The value of positive reward φ^+	1
The value of positive reward φ^-	-1
The value of discount factor λ	0.9
The initial value of exploration rate ε_s	0.99
The end value of exploration rate ε_e	0.001

A. Simulation settings and parameters

In the simulation, we adopted the CPU-based server with version Intel Core i7-8700, 8GB memory. The software environment of DRLVS is TensorFlow1.12.0 with Python 3.6. The trajectories of the vehicles are extracted from UCI GPS trajectory databases [26], which contains 38092 vehicles varying from 2014.9.13 7:24 to 2016.1.19 13:01.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905263, IEEE Access

We are interested in the total STC of selected vehicles, thus this indicator is simulated under the control of different variables like the number of vehicles, sensing rewards and sensing periods. Specifically, the number of vehicles distributed in an range of $[8,34]$, and the sensing reward is between $[10,22]$, and sensing period is in $[5,10]$, which take integers in their own ranges. For all the candidate vehicles, their sensing reward is uniformly distributed in [1, 5]. In our simulation, we extract a subset of vehicles from the dataset for our simulation. The parameters of DRL are presented in Table 2.

B. Numerical results

Fig. 4 shows the convergence procedure of the STC performance along with the increases of training step under different vehicle number with same sensing reward. It should be noted that we take an average value of STC every 100 training steps to constrict amplitude of the training process.

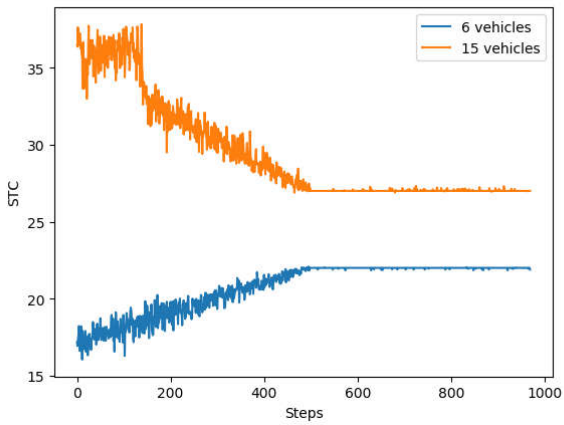

FIGURE 4. The convergence process of STC.

From Fig. 4 we can see that at the beginning of the training process, the STC curve has quite large amplitude with both 15 and 6 candidates because of the initial exploration period. Then after about 50000 steps, the STC converge to certain values, which means by implementing DRLVS scheme, the globally optimal vehicle set is found and its corresponding STC is the achievable max value. Furthermore, subject to the budget of sensing reward, the two curves show different convergence trends.

It is notable that, for 6 vehicles, the STC increases to convergence value, while for 15 vehicles, the STC decreases to convergence value. For 6 vehicles, in the initial exploration phase, its STC is small and the MCS sever has extra budget to recruit vehicles, so after training about 50000 steps, the curve increases to the convergence value. For 15 vehicles, the curve decreases to convergence value. The reason is that in the exploration phase, the STC covered by chosen vehicles is large and the total sensing reward exceed the budget. So the MCS sever keeps exploring and change the choosing policy so that the STC can converge to a value that is maximum under the budget.

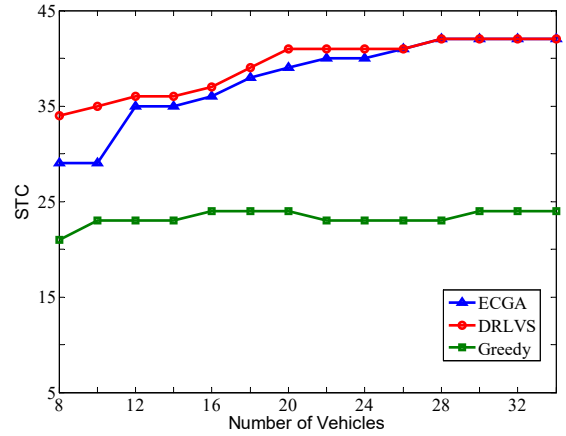

FIGURE 5. STC versus the number of vehicles.

Fig. 5 illustrates the STC performance of different vehicle selection strategies under the different number of vehicles varying from 8 to 34 with budget 10 and sensing period 10. The STC increases with the vehicle number, that is because more vehicles bring in more trajectories and MCS server get more choices to optimize the final selection. However, for each curve, the increase of STC is quite small and when vehicle number reaches a certain value (i.e. 28 in Fig. 4), the STC reaches the upper bound and keeps stable, which fits the reality. We can infer from the simulation results that the number of vehicles have little impact on the STC, due to the limited budget which restricts the MCS server to recruit more vehicles. We can also observe from the figure that our proposed algorithm in this paper outperforms the other algorithms, such as ECQA we previously proposed in [6] and greedy algorithm. The greedy algorithm is not efficient with MCS problem, because it only focuses on satisfying the total sensing reward rather than the STC performance. As to ECQA, it cannot get an optimum solution due to the initial vehicle set, which is chosen by random [6].

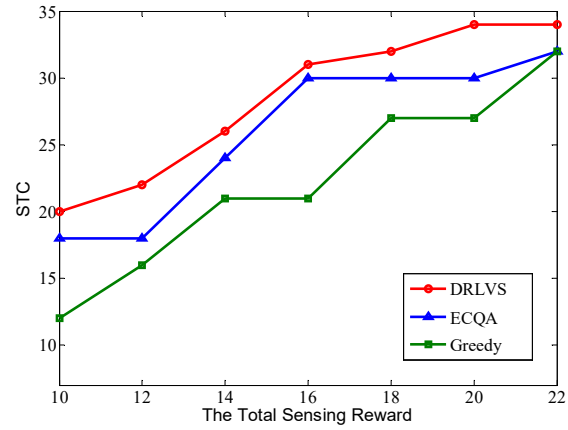

FIGURE 6. STC versus the total sensing reward.

Fig. 6 shows the impact of the total sensing reward, i.e. budget, with 10 vehicles and sensing period 6 on STC. Similarly, STC also increases with the budget. From the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2905263, IEEE Access

increasing trend, we figure out that as long as the budget increases, STC keeps increasing without restriction. Besides, all the curves have an obvious growth on STC. The reason is that when the MCS server possesses more budgets, it recruits more vehicles whose sensing costs are fixed in our model. Our proposed DRLVS algorithm performs better than other algorithms. For greedy algorithm, which only choose the vehicles with lowest sensing rewards, it has no guarantee to the STC performance. For the ECQA algorithm, the factor influences STC performance is the initial vehicle set chosen by random.

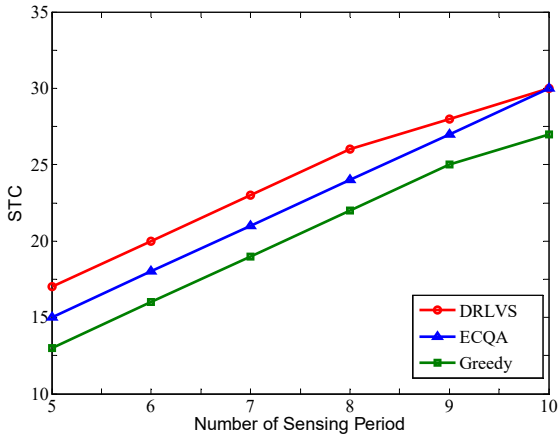

FIGURE 7. STC versus the number of sensing period.

In Fig. 7, with 8 vehicles and sensing rewards 10, the STC performance of different algorithms under different sensing periods are illustrated. As the sensing time increases, the STC increases significantly. Because the sensing time denotes how many ROIs a vehicle can cover when performing sensing tasks. For example, at the beginning, there are 6 vehicles participate in sensing tasks, and every additional sensing period denotes 6 more ROIs to be covered at most. Thus, the increase of STC is large. Comparing with other algorithms, DRLVS can get the optimal solution and reach the highest STC. For the ECQA algorithm, the factor influences STC performance is the initial vehicle set chosen by random.

VI. CONCLUSION

In this paper, we formulate the spatial temporal coverage problem in MCS system as an optimization problem and propose a deep reinforcement learning based vehicle selection scheme to maximize the STC with limited budget. Specifically, the MCS server worked as an agent to interact with the environment and obtain the optimal vehicle selection policy by training the deep Q-network. Simulation results show that our DRLVS scheme is effective to choose vehicles from the candidates and maximize the STC under restrictions.

REFERENCES

[1] M. G. R. Alam *et al*., "Autonomic computation offloading in mobile edge for IoT applications," *Future Generation Computer Systems*, vol. 90, pp. 149-157, Jan. 2019

[2] E. Al Nuaimi *et al.*, "Applications of big data to smart cities," *J. Internet Services Appl.*, vol. 6, p. 25, Dec. 2015.

[3] B. Guo *et al.*, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Comput. Surv.*, vol. 48, no. 1, p. 7, 2015.

[4] Z. He, J. Cao, and X. Liu, "High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility," in *IEEE INFOCOM*, 2015.

[5] L. Xiao *et al*., "A Secure Mobile Crowdsensing Game with Deep Reinforcement Learning," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 35-47, Aug. 2017.

[6] C. Wang *et al*. "Maximizing spatial-temporal coverage in mobile crowd-sensing based on public transports with predictable trajectory," *Int. J. of Distrib. Sens. N*, vol. 14, pp. 1-10, Jul. 2018.

[7] Z. Zhou *et al*. "Robust Mobile Crowd Sensing: When Deep Learning Meets Edge Computing," *IEEE Network* 32.4(2018):54-60.

[8] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Commun. Mag*., vol. 49, no. 11, pp. 32–39, Nov. 2011.

[9] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad, "Mobile phone sensing systems: A survey," *IEEE Commun. Survey Tuts*., vol. 15, no. 1, pp. 403–427, Apr. 2013.

[10] P. Mohan, V. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of Road and Traffic Conditions Using Mobile Smartphones," *Proc. ACM SenSys*, 2008, pp. 323–36.

[11] A. Thiagarajan *et al*., "VTrack: Accurate, energy-aware traffic delay estimation using mobile phones," in *Proc. 7th ACM SenSys*, Nov. 2009, pp. 85–98.

[12] S. B. Eisenman *et al.*, "The Bikenet Mobile Sensing System for Cyclist Experience Mapping," *Proc. SenSys*, Nov. 2007.

[13] A. M. Abu , *et al*., "The Accuracy-Privacy Trade-off of Mobile Crowdsensing." *IEEE Commun. Mag*., 55.6(2017):132-139.

[14] X. Tao *et al*., "Real-time personalized content catering via viewer sentiment feedback: a QoE perspective." *IEEE Network*, 29.6(2015):14-19.

[15] L. Xiao *et al*., "Secure Mobile Crowdsensing Based on Deep Learning," *China communication*, vol. 15, pp. 1-11, Oct. 2018.

[16] S. Gisdakis, T. Giannetsos, and P. Papadimitratos, "Security, privacy, and incentive provision for mobile crowd sensing systems," *J. Internet Things*, vol. 3, no. 5, pp. 839–853, Oct. 2016.

[17] D. Zhao, X.-Y. Li, and H.-D. Ma, "How to Crowdsource Tasks Truthfully Without Sacrificing Utility: Online Incentive Mechanisms with Budget Constraint," *Proc. IEEE INFOCOM*, 2014, pp. 1213–21.

[18] D. Peng, F. Wu, and G. Chen, "Data quality guided incentive mechanism design for crowdsensing," *IEEE Trans. Mobile Comput*., vol. 17, no. 2, pp. 307–319, Feb. 2018.

[19] K. Han, C. Zhang, J. Luo, M. Hu, and B. Veeravalli, "Truthful scheduling mechanisms for powering mobile crowdsensing," *IEEE Trans. Comput*., vol. 65, no. 1, pp. 294–307, Jan. 2016.

[20] B. Guo *et al*., "From Participatory Sensing to Mobile Crowd Sensing," *IEEE PerCom Workshops (SCI)*, 2014.

[21] Y. Zhu, Y. Bao, and B. Li, "On maximizing delay-constrained coverage of urban vehicular networks," *IEEE J. Sel. Areas Commun*., vol. 30, no. 4, pp. 804-817, May 2012.

[22] Y. Liu, J. Niu , and X. Liu, "Comprehensive tempo-spatial data collection in crowd sensing using a heterogeneous sensing vehicle selection method," *Pers. Ubiquit. Comput*., vol. 20 no. 3, pp. 397-411, 2016

[23] V. Mnih *et al*., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, Feb. 2015.

[24] R. S. Sutton and A. G. Barto "Reinforcement Learning: An Introduction," 2nd ed., pp. 71-72. [online] Available: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.

[25] Y. Chen and H. Wang, "IntelligentCrowd: Mobile Crowdsensing via Multi-agent Reinforcement Learning," arXiv:1809.07830v1, Sep. 2018.

[26] M.O. Cruz, H. Macedo, and A. Guimaraes, "Grouping similar trajectories for carpooling purposes," in *Proc. BRACIS*, Natal, Brazil, 2015 pp. 234-239.