Cooperative Task Allocation in Edge Computing Assisted Vehicular Crowdsensing

Yili Jiang[1,3], Kuan Zhang[1], Yi Qian[1], and Rose Qingyang Hu[2]
[1]Department of Electrical and Computer Engineering, University of Nebraska-Lincoln, USA
[2]Department of Electrical and Computer Engineering, Utah State University, USA
[3]College of Communication and Information Engineering, Nanjing University of Posts and Telecommunications, China

***Abstract*—As a popular scenario of mobile crowdsensing, edge computing assisted vehicular crowdsensing (EVCS) encourages vehicles to participate in sensing data with the equipped devices. Due to the vehicular mobility, vehicles may dynamically enter and leave the coverage area of an edge node, leading to recurrent task allocations that consume excessive communication and computational resources. How to avoid recurring recruitment in task allocation is challenging. In this paper, we propose an optimization framework to facilitate task allocation by utilizing the cooperation between edge nodes. The proposed framework avoids complicated recruitment procedures while maximizing the connection time between the recruited vehicles and the edge node. Due to the NP-hardness of the formulated optimization problem, we design a reinforcement learning based algorithm to solve the problem with high accuracy and efficiency. Simulation results show the effectiveness of our proposed framework.**

***Index Terms*—task allocation, cooperation, edge computing, vehicular crowdsening.**

I. Introduction

The ubiquitous smart devices have promoted the mobile crowdsensing, which encourages mobile nodes to participate in collecting data for diverse applications. Vehicular crowdsensing is a special popular scenario of mobile crowdsensing. Equipped with onboard unit (OBU), global positioning system (GPS), cameras, and various sensors, vehicles are able to sense the information of surrounding environment to assist in parking navigation, traffic monitoring and road surface condition inspection [1–3]. In vehicular crowdsensing, a centralized cloud server generally allocates sensing tasks to the qualified vehicles based on their locations, reputations, costs, etc. According to the evaluated quality of the collected data, the recruited vehicles (also know as workers) receive monetary reward from the cloud server. Due to the wide coverage of vehicles, vehicular crowdsensing has superior advantages in terms of financial cost compared with the conventional sensor networks [4–6].

Although vehicular crowdsensing shows great benefits, remote data transmission and centralized data processing are not recommended, since these may lead to longer latency and reduce the system efficiency. To solve the problem, edge computing is used to distribute computation, communication, and storage closer to vehicles [7, 8]. In contrast with the centralized task allocation in conventional vehicular crowdsensing, EVCS allocates tasks at the edge node. In other words, instead of the cloud server, each edge node is responsible for recruiting workers in its coverage area to sense data. However, due to the mobility of vehicles, task allocation has the following two challenges in EVCS: 1) Since vehicles have various velocities, their connection time with edge node is different, affecting data collection. For instance, a vehicle that has longer connection time senses and contributes more data to the task. 2) Efficient task allocation is crucial. Since recruitment generally involves a sequence of steps, such as task release, task competition, reputation verification, etc. Performing these steps leads to longer latency. As the sensed data (e.g., traffic flow data) could be time-sensitive, the latency affects the validity of data.

To tackle these challenges, diverse optimization frameworks for mobility-based task allocations are proposed. Wang *et al.* [9, 10] designed both offline and online algorithms to dynamically maximize the system utility in task allocation, where the mobility of vehicles is represented by the probability of appearing in a specific location. Wu *et al.* [11] proposed a framework which minimizes system latency via introducing mobility prediction. By predicting the location of vehicles, the system can promote proactive task allocation, reducing the computation latency. Since the mobility model only depends on the vehicle's acceleration, the prediction may not be accurate. Li *et al.* [12] focused on maximizing the system utility through prediction-based dynamic task allocation. Semi-Markov model is used to predict the transition probability of a worker from a position to another position. However, since the above frameworks perform the task allocation within independent edge nodes, vehicles may dynamically enter and leave the coverage area of an edge node, leading to rapid changes in the vehicular network topology. When a worker leaves, the edge node needs to recruit new workers, resulting in recurrent task allocations with excessive resource consumption and longer latency.

In this paper, we propose a novel scheme to avoid recurrent task allocation by exploring cooperative benefits of edge nodes. Specifically, the edge nodes can cooperatively exchange the mobility information and work status of vehicles between each other. Then an edge node E can be informed by the availability of vehicles in the next time interval. If these vehicles are workers from other edge nodes, E can assign tasks to them without processing all recruitment steps (e.g., task release, reputation calculation). Then the procedures of task allocation can be facilitated, reducing resources consumption

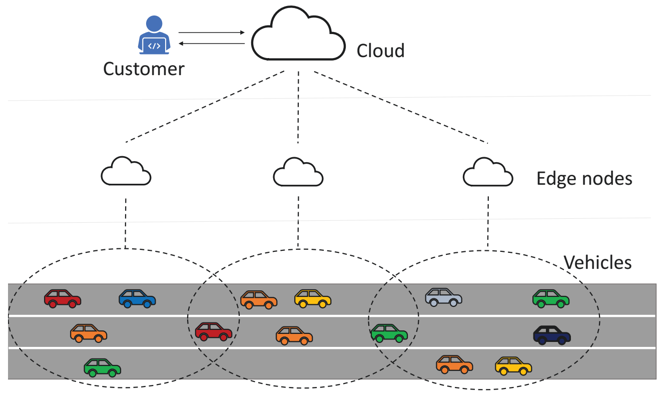


Fig. 1. System model.

and system latency. Considering such cooperative task allocation, we propose a framework which maximizes the connection time while avoiding to perform all recruitment procedures. The main contributions of this paper are summarized as follows.

- Utilizing the cooperation between edge nodes, we propose an optimization framework for task allocation in EVCS. The proposed framework maximizes the workers' connection time with edge nodes and avoids processing all recruitment steps.
- Due to the NP-hardness of the formulated problem, we propose to relax the optimization problem into two subproblems. A reinforcement learning based algorithm is designed to solve the transformed problems efficiently.
- We conduct simulation experiments to evaluate our scheme. The performance evaluation indicates that the proposed framework can optimize task allocation with high accuracy and efficiency.

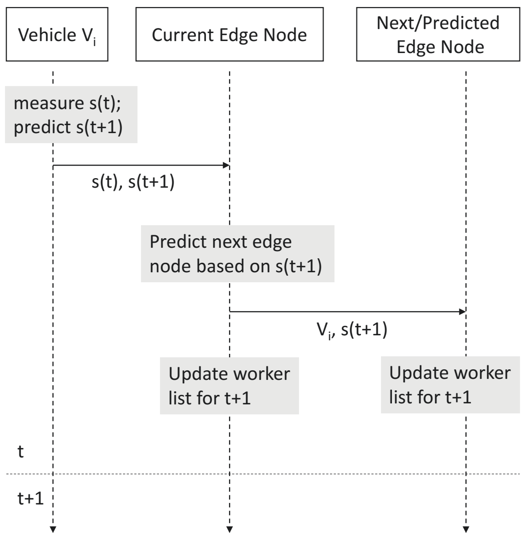
The rest of this paper is organized as follows. In Section II, the system model is provided. In Section III, the problem statement is described. After that, the proposed approaches are discussed in Section IV. Simulation results are evaluated in Section V and conclusions are provided in Section VI.

II. System Model

A. System Overview

Four types of entities (customer, cloud server, edge nodes, and vehicles) are generally involved in the EVCS system shown in Fig. 1. The detailed descriptions of each type of entity are presented as follows.

- *Customer:* A customer is a service requestor, which aims to collect data from some specific locations and perform data analysis or data mining. A customer can be an individual or a company. For example, a company may need to collect real-time data of road surface to monitor the road surface condition and schedule maintenance accordingly. However, since a customer generally has limited or constrained communication and computation resources, it is unable to finish the task by itself. Thus, the customer outsources such a sensing task to the cloud server.



Vehicle V_i
Current Edge Node
Next/Predicted Edge Node
measure s(t); predict s(t+1)
s(t), s(t+1)
Predict next edge node based on s(t+1)
V_i, s(t+1)
Update worker list for t+1
Update worker list for t+1
t
t+1

Fig. 2. Procedure of cooperative task allocation in EVCS.

- *Cloud Server:* The cloud server is the service provider for customers. It releases the task to the edge nodes whose coverage is in the task area. Assisted by the edge nodes, the cloud server obtains data collected from the task area. By performing data mining and information aggregation, the cloud server returns the final results to the customer.
- *Edge Nodes:* As part of service provider, edge nodes provide assistance to the cloud server in task allocation and data transmission. Specifically, in its coverage, each edge node is responsible for recruiting vehicles to collect data and transmitting the extracted information to cloud server. Edge nodes are close to vehicles and have sufficient resources for communication and computation.
- *Vehicles:* Equipped with OBU and other smart devices, vehicles are able to sense data from their surroundings. If a vehicle is interested in the task, it sends its information (e.g., location, reputation, expected reward, etc.) to the edge node to compete with other candidates. The candidates with good reputation have high probability to be selected. The recruited vehicles are referred as workers.

Without loss of generality, nine steps are considered in such a EVCS system. Complete task allocation involves Steps 2-5.

Step 1: The customer sends a task requirement and budget information to the cloud server.

Step 2: The cloud server releases the task to the corresponding edge nodes based on the location of task area.

Step 3: Each edge node broadcasts the task to the vehicles within its coverage.

Step 4: Vehicles upload their information to the edge node to compete the task.

Step 5: The edge node recruits qualified vehicles and assigns the task accordingly.

Step 6: The vehicles sense data from the environment and submit the collected data to the edge node.

TABLE I
NOTATION DEFINITIONS

Variable	Definition
$\boldsymbol{O}(t)$	Vehicular status
(p_x, p_y)	Position information
(v_x, v_y)	Velocity information
(a_x, a_y)	Acceleration information
(u_x, u_y)	Noise coefficients
$\mathcal{E}$	Set of edge nodes
E_n	Edge node n
$\mathcal{V}_n(t)$	Set of vehicles
V_i	Vehicle i
$\mathcal{D}_n(t)$	Set of vehicles that are workers at time $t-1$
$\mathcal{G}_n(t)$	Set of vehicles that are not workers at time $t-1$
$C_i(t)$	Reward cost of V_i
$T_i(t)$	Connection time of V_i
δ	Reward limitation
K	Number of required workers
$\rho_i(t)$	$\rho_i(t)=1$, if V_i is recruited; $\rho_i(t)=0$, otherwise.
Q	Large constant
$\mu(t)$	$\mu(t)=1$, if all recruitment steps are required to process; $\mu(t)=0$, otherwise.

Step 7: The edge node extracts information and sends aggregated data to the cloud server.

Step 8: The cloud server aggregates the final report and returns to the customer.

Step 9: The cloud server issues rewards to the edge nodes according to the data quality. The edge nodes distribute rewards to the corresponding vehicles.

B. Mobility Prediction

Equipped with GPS and sensors, a vehicle can measure its location, velocity, and acceleration. Within a 2D surface, we define the status of a vehicle as $\boldsymbol{O} = (p_x, p_y, v_x, v_y, a_x, a_y)$, where (p_x, p_y), (v_x, v_y), (a_x, a_y) are the components of location, velocity, and acceleration in directions x and y respectively. Then we can use the status at time t to predict the status at time $t+1$ following

$$\boldsymbol{O}(t+1) = \boldsymbol{A}\boldsymbol{O}(t), \tag{1}$$

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1+u_x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1+u_y \end{bmatrix}, \boldsymbol{O}(t) = \begin{bmatrix} p_x(t) \\ p_y(t) \\ v_x(t) \\ v_y(t) \\ a_x(t) \\ a_y(t) \end{bmatrix},$$

where $\boldsymbol{A}$ is the transitional matrix and Δ is the size of the time interval. u_x and u_y represent the noise coefficients. All notations used in this paper are listed in Table I.

III. PROBLEM FORMULATION

Fig. 2 shows the procedure of cooperative task allocation in the edge node. At time t, each vehicle measures its status $\boldsymbol{O}(t)$ with equipped devices and predicts the status $\boldsymbol{O}(t+1)$ following Eq. (1). Then $\boldsymbol{O}(t)$ and $\boldsymbol{O}(t+1)$ are sent to the current edge node. Based on the position information $(p_x(t+1), p_y(t+1))$ in $\boldsymbol{O}(t+1)$, the edge node can predict the next edge node that the vehicle is going to connect at time $t+1$. Practically, the predicted edge node is either the current edge node or an adjacent edge node. After that, the current edge node sends the vehicle's information (e.g., identity, work status) and $\boldsymbol{O}(t+1)$ to the predicted edge node. By doing this, each edge node knows details of vehicles which are going to leave or enter the coverage area. Then the edge node performs proactive task allocation for the next time interval. As discussed in the previous section, if the workers have longer connection time with the edge node, they contribute more data to the task. In addition, recruiting previous workers can avoid performing all recruitment steps. Therefore, how to select workers to maximize their connection time while avoiding performing all recruitment steps is the targeted problem in this paper.

We consider a set of edge nodes $\mathcal{E} = \{E_1, E_2, ..., E_n, ...\}$. Each edge node E_n involves a set of vehicles $\mathcal{V}_n(t) = \{V_1, V_2, ..., V_{M_n(t)}\}$ at time t. $\mathcal{V}_n(t)$ can be further divided into two sets $\mathcal{D}_n(t)$ and $\mathcal{G}_n(t)$, where $\mathcal{V}_n(t) = \mathcal{D}_n(t) \cup \mathcal{G}_n(t)$, $\mathcal{D}_n(t) \cap \mathcal{G}_n(t) = \emptyset$. $\mathcal{D}_n(t)$ stands for the vehicles that are workers at time $t-1$. Note that these workers can be either under the current edge node or an adjacent edge node at time $t-1$. $\mathcal{G}_n(t)$ denotes the vehicle that does not work at time $t-1$. Similarly, these vehicles can be either under the current edge node or an adjacent edge node at time $t-1$. For each vehicle V_i, we introduce a binary parameter $\rho_i(t)$. If V_i is selected as a worker at time t, $\rho_i(t) = 1$. Otherwise, $\rho_i(t) = 0$. The reward cost and predicted connection time of V_i at time t are denoted as $C_i(t)$ and $T_i(t)$. $\mu(t)$ denotes if all recruiting steps are required. To maximize the connection time while facilitating recruitment for each edge node at time t, we formulate the optimization problem as follows.

$$\mathbf{P1}: \text{Max} \frac{\sum\limits_{\forall i} T_i(t)\rho_i(t)}{\mu(t)+1} \tag{2}$$

$$\text{s.t.} \quad \sum_i C_i(t)\rho_i(t) \le \delta, \quad \forall V_i \in \mathcal{V}_n(t), \tag{3}$$

$$\sum_i \rho_i(t) \ge K, \quad \forall V_i \in \mathcal{V}_n(t), \tag{4}$$

$$\sum_i \rho_i(t) \le \mu(t) Q, \quad \forall V_i \in \mathcal{G}_n(t), \tag{5}$$

$$\rho_i(t) \in \{0,1\}, \quad \forall V_i \in \mathcal{V}_n(t), \tag{6}$$

$$\mu(t) \in \{0,1\}, \tag{7}$$

where δ is the reward budget of the edge node. (3) represents the budget limitation. (4) is the constraint of worker number. To collect sufficient data, the number of recruited workers cannot be less than K. In (5), Q is a large constant that forces $\mu(t)=0$ when $\sum_i \rho_i(t) = 0, \forall V_i \in \mathcal{G}_n(t)$ and $\mu(t)=1$ when $\sum_i \rho_i(t) > 0, \forall V_i \in \mathcal{G}_n(t)$. It indicates that when the edge node selects new workers from $\mathcal{G}_n(t)$ at time t, all recruitment steps are required to process. Since $\mu(t)$ is possible to be 0, $\mu(t)+1$ is used force the denominator to be positive in (2).

Theorem 1: $\mathbf{P1}$ is NP-hard.

Proof: $\mathbf{P1}$ is a non-linear integer programming problem. By setting $\mu(t) = 1$ and $K = 0$, $\mathbf{P1}$ is degenerated to $0-1$ knapsack problem, which is NP-hard. Therefore, $\mathbf{P1}$ is NP-hard. ■

IV. Proposed Approaches

Since $\mathbf{P1}$ is NP-hard, it is infeasible to find optimal solution in polynomial time. We are motivated to solve $\mathbf{P1}$ within two stages: 1) As discussed previously, $\mu(t) = 1$ means all recruitment steps are required to process given that a worker is selected from $\mathcal{G}_n(t)$. To avoid recurrent task allocation, we prefer to recruit workers from $\mathcal{D}_n(t)$ first without considering constraint (4). 2) If the number of the recruited workers satisfies constraint (4), we use the corresponding recruitment result as the suboptimal solution to $\mathbf{P1}$. Otherwise, we consider to recruit more workers from $\mathcal{G}_n(t)$.

To implement the two stages, we divide $\mathbf{P1}$ into two subproblems depending on the value of $\mu(t)$. Given $\mu(t) = 0$, $\mathbf{P1}$ is transformed into a simplified linear integer programming as follows,

$$\mathbf{P2}: \text{Max} \sum_{\forall i} T_i(t)\rho_i(t) \tag{8}$$

$$\text{s.t.} \quad \sum_i C_i(t)\rho_i(t) \le \delta, \quad \forall V_i \in \mathcal{D}_n(t), \tag{9}$$

$$\sum_i \rho_i(t) \ge K, \quad \forall V_i \in \mathcal{D}_n(t), \tag{10}$$

$$\rho_i(t) \in \{0, 1\}, \quad \forall V_i \in \mathcal{D}_n(t).$$

Given $\mu(t) = 1$, $\mathbf{P1}$ can be relaxed to

$$\mathbf{P3}: \text{Max} \frac{1}{2} \sum_{\forall i} T_i(t)\rho_i(t) \tag{11}$$

$$\text{s.t.} \quad \sum_i C_i(t)\rho_i(t) \le \delta, \quad \forall V_i \in \mathcal{V}_n(t), \tag{12}$$

$$\sum_i \rho_i(t) \ge K, \quad \forall V_i \in \mathcal{V}_n(t), \tag{13}$$

$$\rho_i(t) = 1, \quad \forall V_i \in \mathcal{D}_n(t), \tag{14}$$

$$\rho_i(t) \in \{0, 1\}, \quad \forall V_i \in \mathcal{V}_n(t). \tag{15}$$

Theorem 2: $\mathbf{P2}$ and $\mathbf{P3}$ are NP-hard.

Proof: By setting $K = 0$, $\mathbf{P2}$ and $\mathbf{P3}$ can be degenerated to $0-1$ knapsack problem, which is NP-hard. Therefore, $\mathbf{P2}$ and $\mathbf{P3}$ are both NP-hard. ■

$\mathbf{P2}$ and $\mathbf{P3}$ represent the optimization problems in stage 1) and stage 2) respectively. Then following the above two stages, we first solve $\mathbf{P2}$, if we can find a feasible solution for $\mathbf{P2}$, we use that solution for $\mathbf{P1}$. Otherwise, we continue to solve $\mathbf{P3}$ and use the solution of $\mathbf{P3}$ for $\mathbf{P1}$. Due to the hardness of $\mathbf{P2}$ and $\mathbf{P3}$, we are motivated to design an algorithm of reinforcement learning based task allocation (RLTA) to find suboptimal solutions for the problems. Specifically, we employ Q-learning, which is a well-established method in the class of reinforcement learning, in the proposed algorithm. To perform Q-learning algorithm, we model the problem with state space, action space, reward function, and Q-function as follows.

- *Action Space $\mathcal{A}$:* In $\mathbf{P2}$ and $\mathbf{P3}$, if we only consider constraints (9) and (12), then the problem becomes to pick multiple candidates one by one within the reward budget while maximizing their connection time. For each candidate, we can take three actions, involving a_1: taking the candidate as a worker and adding into the worker list directly; a_2: taking the candidate as a worker and replace an existing worker on the worker list to improve the connection time; a_3: not taking the candidate. These three actions form the action space $\mathcal{A}$.
- *State Space $\mathcal{S}$:* For each candidate, since we can take different actions, the temporary worker list may involve various combinations of candidates. We calculate the ratio of connection time to cost for each vehicle that is on the temporary worker list. The vehicle that has the lowest ratio is denoted as V_q. The connection time and reward cost of V_q are T_q and C_q. The present candidate is denoted as V_i. Similarly, the connection time and reward cost of V_i are T_i and C_i. Assuming the residual budget is δ', we use three parameters X, Y, Z to represent different states as following

$$X = \begin{cases} 0 & C_p + \delta' \ge C_i \\ 1 & \text{Otherwise,} \end{cases} \tag{16}$$

$$Y = \begin{cases} 0 & C_p \ge C_i \\ 1 & \text{Otherwise,} \end{cases} \tag{17}$$

$$Z = \begin{cases} 0 & V_i \ge V_p \\ 1 & \text{Otherwise.} \end{cases} \tag{18}$$

- *Reward Function:* At each state, taking an action receives the immediate reward following the reward function

$$r = \begin{cases} 0 & a_1 \text{ is true} \\ \tau^- & a_2 \text{ is true}, C_i > \delta' \\ \tau^+ & a_2 \text{ is true}, C_i \le \delta' \\ \tau^- & a_3 \text{ is true}, C_i \le \delta' \\ \tau^- & a_3 \text{ is true}, C_i > \delta' + C_p \\ \tau^- & a_3 \text{ is true}, V_i < V_p \\ \tau^+ & \text{Otherwise,} \end{cases} \tag{19}$$

  where τ^- is penalty and τ^+ is positive reward.
- *Q-Function:* Q-function is defined to represent the expected accumulative rewards when action a is taken in state s. In this work, the Q-function is derived from Bellman equation as following

$$Q_{t+1}(s, a) = Q_t(s, a) + \beta \left(r + \gamma \max_{a\prime} Q_t(s\prime, a\prime) - Q_t(s, a) \right). \tag{20}$$

Algorithm 1 depicts the learning procedure of RLTA. In each iteration, we first initialize the worker list $\mathcal{W}$, residual budget δ', and state s. For each candidate, we take actions

Algorithm 1 The Learning of RLTA

```
Input: $\mathcal{S}$, $\mathcal{A}$, $\beta$, $\gamma$, $\epsilon$, $\delta$.
Output: $Q$-table.
Initialization: $Q(s, a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$.
1:  for $i = 1, 2, ...$ do
2:      $\mathcal{W} = \emptyset$;
3:      $\delta' = \delta$;
4:      $s \leftarrow \mathcal{S}(X, Y, Z)$;
5:      for each candidate do
6:          Generate random number $e$.
7:          if $e \geq \epsilon$ then
8:              $a \leftarrow \arg\max_{a'} Q(s, a')$.
9:          else
10:             Randomly pick $a \in \mathcal{A}$.
11:         end if
12:         Calculate immediate reward based on (19);
13:         Update $Q(s, a)$ based on (20);
14:         Calculate $X, Y, Z$ based on (16) (17) (18);
15:         $s \leftarrow \mathcal{S}(X, Y, Z)$;
16:         Update $\mathcal{W}$;
17:         Update $\delta'$.
18:     end for
19: end for
20: return $Q$-table.
```

Algorithm 2 The Execution of RLTA

```
Input: $Q$-table, $K$, $\delta$, $\mathcal{D}_n(t)$, $\mathcal{G}_n(t)$.
Output: $\mathcal{W}$.
Initialization: $\mathcal{W} = \emptyset$, $\delta' = \delta$.
1:  for $V_i \in \mathcal{D}_n(t)$ do
2:      Calculate $X, Y, Z$ based on (16) (17) (18);
3:      $s \leftarrow \mathcal{S}(X, Y, Z)$;
4:      $a \leftarrow \arg\max_{a'} Q(s, a')$;
5:      Update $\mathcal{W}$;
6:      Update $\delta'$.
7:  end for
8:  if len($\mathcal{W}^Q$) $\geq K$ then
9:      return $\mathcal{W}$.
10: else
11:     for $V_i \in \mathcal{G}_n(t)$ do
12:         Calculate $X, Y, Z$ based on (16) (17) (18);
13:         $s \leftarrow \mathcal{S}(X, Y, Z)$;
14:         $a \leftarrow \arg\max_{a'} Q(s, a')$;
15:         Update $\mathcal{W}$;
16:         Update $\delta'$.
17:     end for
18:     return $\mathcal{W}$.
19: end if
```

under ϵ-greedy policy to balance exploration and exploitation. Specifically, the agent can take a random action with probability ϵ or take the best action in the given state with probability $1 - \epsilon$. The best action is derived by searching the maximum value in the Q-table given the present state. After that, the immediate reward for the selected action is obtained by calculating the reward function (19). Based on the immediate reward, we can update the Q-table following the Q-function (20). Subsequently, by updating $\mathcal{W}$ and δ', the environment turns to the next state. With sufficient iterations, the Q-table is well-established.

Algorithm 2 summarizes the execution of RLTA. As discussed previously, we are motivated to solve $\mathbf{P2}$ first. For each candidate in the set $\mathcal{D}_n(t)$, we can get the present state through calculating X, Y, Z. By searching the Q-table, the best action for the current state is selected. Then based on the selected action, we may add new worker to $\mathcal{W}$. We keep updating $\mathcal{W}$ until all candidates in $\mathcal{D}_n(t)$ are considered. At the end, we obtain the final $\mathcal{W}$, which is the solution for $\mathbf{P2}$. If the number of workers in $\mathcal{W}$ is greater than K, we return $\mathcal{W}$ as the solution for $\mathbf{P1}$. Otherwise, we continue to solve $\mathbf{P3}$ following the lines 11-18 in Algorithm 2. Similarly, for all candidates in the set $\mathcal{G}_n(t)$, we can derive the best actions for them by searching Q-table. Thus new workers can be selected from $\mathcal{G}_n(t)$. At the end, the $\mathcal{W}$ is updated accordingly and returned as the solution for $\mathbf{P1}$.

V. Performance Evaluations

In the simulation results presented in this section, we consider a network with one cloud and ten edge nodes. The communication range of an edge node is 400 meters. In the system, the number of vehicles varies within [50,150] at different time intervals. The velocity of each vehicle is randomly set from 20 to 45 miles/hour. Due to the vehicular mobility, the connection time between the vehicles and the edge node are various. We compare the proposed algorithm with [12] (where a greedy algorithm is proposed). In addition, we evaluate a heuristic algorithm, which tests all possible combinations of candidates to find the best solution. Since this algorithm tests all possible solutions, we use its performance as the benchmark to estimate our proposed algorithm. The simulation results show the performance under the comparison of the three methods. Each point in the following figures is based on the average values of 1000 simulation runs.

Fig. 3 shows the vehicles' connection time at each edge node. With the increasing of time intervals, although all the three methods obtain relatively stable connection time, our proposed algorithm outperforms the greedy algorithm significantly with a higher value of connection time. Compared with the heuristic algorithm, our proposed algorithm degrades slightly. Fig. 4 quantifies the performance. Since the heuristic algorithm is the benchmark, we set its optimization accuracy as 100%. Comparing with the benchmark, our proposed algorithm achieves high accuracy at around 95%. It means that our proposed algorithm is effective to find the optimal solution.

Fig. 5 describes the occurrence of reccruent task allocation, indicating the number of times that all recruitment steps are processed over time. In greedy algorithm, with the increasing of time interval, the number of times is increased linearly. The performance of our proposed algorithms is much better than the greedy algorithm benefiting from the cooperation of

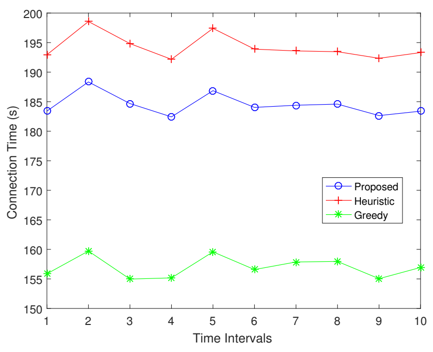


Fig. 3. Optimized connection time.

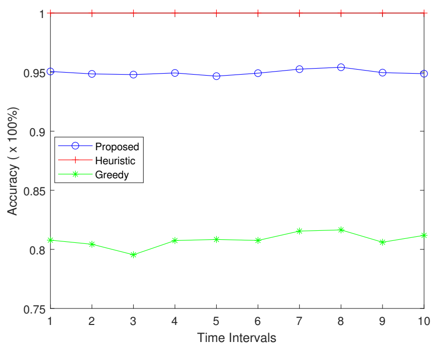


Fig. 4. Accuracy over time.

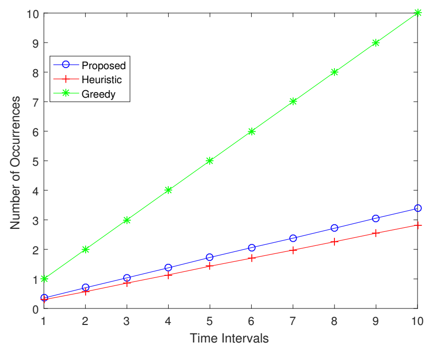


Fig. 5. Occurrence of recurrent task allocation.

edge nodes. Although the heuristic algorithm slightly outperforms our proposed algorithm, the proposed algorithm has less computational complexity. As shown in Algorithm 2, the computational complexity is $\mathcal{O}(N)$, where N is the number of candidates. For the heuristic algorithm, its computational complexity is $\mathcal{O}(N^2)$. Overall, our proposed algorithm can solve the problem accurately with higher efficiency.

VI. Conclusion

In this paper, we have proposed an optimization framework for cooperative task allocation in EVCS, where the edge nodes collaborate to exchange vehicles' mobility information and recruitment status. The proposed framework aims to avoid performing all recruitment steps in task allocation and maximize the connection time of workers. A non-linear integer programming problem has been formulated to describe the optimization goals. We have designed a reinforcement learning based algorithm to solve the optimization problem. The simulation results have demonstrated that our proposed framework achieves the optimization goals with high accuracy and efficiency. In future work, we will investigate the accuracy of mobility prediction and its relationship with recurrent recruitment in the proposed cooperative task allocation.

Acknowledgment

This work was partially supported by National Science Foundation under grants CNS-2007995 and CNS-2008145.

References

[1] C. Wang, Z. Xie, L. Shao, Z. Zhang, and M. Zhou, "Estimating travel speed of a road section through sparse crowdsensing data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3486–3495, 2019.

[2] S. Basudan, X. Lin, and K. Sankaranarayanan, "A privacy-preserving vehicular crowdsensing-based road surface condition monitoring system using fog computing," *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 772–782, 2017.

[3] S. Abdul Rahman, A. Mourad, and M. El Barachi, "An infrastructure-assisted crowdsensing approach for on-demand traffic condition estimation," *IEEE Access*, vol. 7, pp. 163 323–163 340, 2019.

[4] L. Xiao, T. Chen, C. Xie, H. Dai, and H. V. Poor, "Mobile crowdsensing games in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 2, pp. 1535–1545, 2018.

[5] W. Guo, W. Zhu, Z. Yu, J. Wang, and B. Guo, "A survey of task allocation: contrastive perspectives from wireless sensor networks and mobile crowdsensing," *IEEE Access*, vol. 7, pp. 78 406–78 420, 2019.

[6] J. Guo, B. Song, Y. He, F. R. Yu, and M. Sookhak, "A survey on compressed sensing in vehicular infotainment systems," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2662–2680, 2017.

[7] J. Li, Z. Su, D. Guo, K.-K. R. Choo, Y. Ji, and H. Pu, "Secure data deduplication protocol for edge-assisted mobile crowdsensing services," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 742–753, 2021.

[8] J. Ni, A. Zhang, X. Lin, and X. S. Shen, "Security, privacy, and fairness in fog-based vehicular crowdsensing," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 146–152, 2017.

[9] X. Wang, R. Jia, X. Tian, and X. Gan, "Dynamic task assignment in crowdsensing with location awareness and location diversity," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 2420–2428.

[10] X. Wang, R. Jia, X. Tian, X. Gan, L. Fu, and X. Wang, "Location-aware crowdsensing: Dynamic task assignment and truth inference," *IEEE Transactions on Mobile Computing*, vol. 19, no. 2, pp. 362–375, 2020.

[11] X. Wu, S. Zhao, R. Zhang, and L. Yang, "Mobility prediction-based joint task assignment and resource allocation in vehicular fog computing," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–6.

[12] D. Li, J. Zhu, and Y. Cui, "Prediction-based task allocation in mobile crowdsensing," in *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, 2019, pp. 89–94.