Regret-Matching Learning-Based Task Assignment in Vehicular Edge Computing

Bach Long Nguyen, Duong D. Nguyen, Hung X. Nguyen, and Duy T. Ngo

Abstract—Vehicular edge computing has recently been proposed to support computation-intensive applications in Intelligent Transportation Systems (ITS) such as augmented reality and autonomous driving. Despite recent progress in this area, significant challenges remain to efficiently allocate limited computation resources to a range of time-critical ITS tasks. Toward this end, the current paper develops a new task assignment scheme for vehicles in a highway scenario. Because of the high speed of vehicles and the limited communication range of roadside units (RSUs), the computation tasks of participating vehicles are to be migrated across multiple servers. We formulate a binary nonlinear programming (BNLP) problem of assigning computation tasks from vehicles to RSUs and a macrocell base station. To deal with the potentially large size of the formulated optimization problem, we develop a distributed multi-agent regret-matching learning algorithm. Based on the regret minimization principle, the proposed algorithm employs a forgetting method that allows the learning process to quickly adapt to and effectively handle the high mobility feature of vehicle networks. We theoretically prove that it converges to the correlated equilibrium solutions of the considered BNLP problem. Simulation results with practical parameter settings show that the proposed algorithm offers the lowest total delay and cost of processing tasks. Importantly, our algorithm converges much faster than existing methods as the problem size grows, demonstrating its clear advantage in large-scale vehicular networks.

Index Terms—Correlated equilibrium, multi-agent learning, regret matching, task assignment, vehicular edge computing

I. INTRODUCTION

Due to limited computation and storage capabilities of vehicular users, it is rather difficult to meet the strict requirements of Intelligent Transportation System (ITS) applications, i.e., intensive computation and content caching with low latency [1]. To this end, Vehicular Edge Computing (VEC), as an application of Mobile Edge Computing (MEC) in high mobility environments, has been proposed as a solution [2]. Even so, there remains a significant challenge to efficiently allocate the limited communication and computation resources of servers in VEC, due to an increasing number of vehicles that need their tasks processed.

Traditional optimization methods for resource allocation require knowledge of channel conditions which are time-varying and, at times, unavailable in high mobility scenarios. These solutions are typically based on a snapshot model of the vehicular networks, while ignoring the long-term influence of the current decision [2]. On the other hand, without any knowledge of the operating environment, reinforcement learning (RL) is able to make decisions that maximize the long-term rewards for the networks. It is arguably a promising tool to tackle problems encountered in task offloading, and communication and computation resource allocation in VEC-based ITS with time varying an unknown channel conditions.

In [3], multiple in-car applications employ an RL-based scheduling strategy to offload their tasks to MEC servers located within road side units (RSUs). Here, the latency and energy consumption for task processing are minimized. Taking a step further, a joint management scheme of spectrum, computing and storing resources in VEC is proposed in [4] using deep reinforcement learning (DRL). Note that in [3] and [4], vehicles potentially reside within the coverage range of RSUs for a short time duration only, due to their high mobility and the limited communication range of the serving RSUs. It is possible that a vehicle already moves out of the range of its serving RSU even before its tasks are completely processed.

The above issue can be overcome by allowing the vehicle to migrate its tasks to the MEC servers of the next RSUs that the vehicle is about to move into. Assuming these MEC servers have large computation resources, [5] uses DRL to minimize the energy consumption for task processing while meeting latency requirements. Also based on DRL, [6] not only develops a task migration scheme but also finds the best migration routes for vehicles. Here, a vehicle only migrates its tasks to an MEC server if the time it takes that vehicle to reach such a server is the shortest.

A major issue with the DRL approaches in [5] and [6] is that a significant training time is required and the algorithm’s convergence is not guaranteed. To address this issue, [7] and [8] employ regret matching (RM) learning to obtain correlated-equilibrium solutions. Aiming for throughput fairness in wireless access networks, the algorithms in [7] and [8], however, are not specifically designed for task migration in VEC. The inherited characteristics of vehicle networks with high mobility may render these solutions inapplicable.

In this paper, we propose an RM learning-based task assignment scheme that minimizes the total delay and cost incurred by vehicles in a highway scenario. We assume that once a vehicle leaves the coverage area of its serving RSU, it will migrate its tasks to other suitable RSUs or a macrocell base station according to its mobility pattern. The contributions of this paper are summarized as follows.
1) Unlike [3] and [4], we formulate a task assignment problem as a binary nonlinear programming (BNLP) problem with specific constraints on the movement of participating vehicles.

2) Unlike [5] and [6], we reformulate the BNLP problem as a standard repeated game. Then, we propose a distributed RM algorithm that decomposes the state observations and actions of a monolithic centralized agent into those of multiple agents. We further propose a forgetting method to speed up the convergence of the traditional RM algorithms in [7], [8]. Doing so helps the algorithm to effectively handle the high level of user mobility in vehicle networks.

3) Our simulation results with practical parameter settings demonstrate the advantages of our solution in terms of delay and cost minimization, and convergence speed particularly in large-scale network settings.

II. SYSTEM MODEL AND PROBLEM FORMULATION

A. Network Scenario

We consider a network scenario in Fig. 1 where vehicles left-drive in two directions along a six-lane highway. A macrocell base station (BS) is deployed to provide network connectivity along the highway. For data and computation offloading, a set $\mathcal{R}$ of RSUs are deployed at an inter-RSU distance of $D_R$ to bring the network closer to the vehicles. We denote by $\{\text{BS}\}$ the set containing only the BS, and by $|\mathcal{R}|$ the number of RSUs. The BS and RSUs are connected via wired links for load balancing and control coordination. Each of them is equipped with a server comprising a data processing unit and a cache.

Assume the vehicles have to complete computation-intensive tasks. Due to their limited computing resources, it is sensible to offload these tasks to the servers at the BS and/or the RSUs. The offload requests are sent via vehicle-to-infrastructure (V2I) communication, which is supported by the Long-Term Evolution-Advanced (LTE-A) protocol. Denote by $\mathcal{J}$ the set of requesting vehicles, and assume a vehicle only requests to offload one task at a time. As such, we refer to vehicle $i \in \mathcal{J}$ and task $t \in \mathcal{J}$ interchangeably. Also, the number of tasks to be offloaded is equal to the number of requesting vehicles.

If a vehicle $i \in \mathcal{J}$ traveling at a constant speed of $v_i$ is still within the coverage range of an RSU $r \in \mathcal{R}$, its offload request is sent directly to the RSU $r$; otherwise, to the BS. In either case, the server at the BS collects from all the RSUs information about task sizes, server computing capabilities, and current location and speed of vehicles. It then computes and makes a task assignment decision as to where the tasks are to be processed, i.e., at the BS or the RSUs; and in the latter case, which RSU in $\mathcal{R}$.

B. Communication Model

We consider that the received signal strength at the RSUs and BS depends only on the positional shift of the vehicles, where the effect of small-scale fading is averaged out. For interference cancellation, we adopt the orthogonal frequency-division multiplexing (OFDM) to assign orthogonal frequencies to the link between an RSU/BS $r \in \mathcal{R} \cup \{\text{BS}\}$ and a vehicle $i \in \mathcal{J}$. The data rate at which the tasks of the vehicle $i$ are uploaded to the RSU/BS $r$ at a given time $t$ is expressed as:

$$R_{r,i}(t) = B_{r,i}(t) \log_2 \left(1 + \frac{p_i|h_{r,i}(t)|^2}{N_r^2}\right), \quad \forall i \in \mathcal{J}, \quad \forall r \in \mathcal{R} \cup \{\text{BS}\}.$$  (1)

where $B_{r,i}(t)$ is the link’s bandwidth, $p_i$ is the transmit power of the vehicle $i$, $|h_{r,i}(t)|^2$ is the link gain between the vehicle $i$ and the RSU/BS $r$, and $N_r^2$ is the received noise power. Here, $|h_{r,i}(t)|^2 = f(d_{r,i}(t))$ with $f(\cdot)$ a path-loss function, and $d_{r,i}(t)$ the Euclidean distance between the vehicle $i$ and the RSU/BS $r$ at the time $t$.

C. Computation Model

The amount of time needed for a task $i \in \mathcal{J}$ to be uploaded to an RSU/BS $r \in \mathcal{R} \cup \{\text{BS}\}$ is given by:

$$T^u_{r,i}(t) = \frac{s_i}{R_{r,i}(t)}, \quad \forall i \in \mathcal{J}, \quad \forall r \in \mathcal{R} \cup \{\text{BS}\}.$$  (2)

where $s_i$ is the size of the task $i$.

We use two binary variables $x_{r,i}(t)$ and $x_{r,\hat{r},i}(t)$ to decide where the task $i \in \mathcal{J}$ is executed at the time $t$. If the task $i$ is to be processed at an RSU/BS $r \in \mathcal{R} \cup \{\text{BS}\}$, then $x_{r,i}(t) = 1$; otherwise, $x_{r,i}(t) = 0$. If the task $i$ is migrated and processed at the other RSU/BS $\hat{r} \in \mathcal{R} \cup \{\text{BS}\} \setminus \{r\}$, then $x_{r,\hat{r},i}(t) = 1$; otherwise, $x_{r,\hat{r},i}(t) = 0$. The task migration time is calculated as [6]:

$$T^m_{r,\hat{r},i}(t) = x_{r,\hat{r},i}(t) \left(\frac{s_i}{B_R} + 2 \cdot \alpha \cdot h_{r,\hat{r}}\right), \quad \forall i \in \mathcal{J}, \quad \forall r, \hat{r} \in \mathcal{R} \cup \{\text{BS}\} \setminus \{r\}.$$  (3)

where $B_R$ is the bandwidth of the wired link between $r$ and $\hat{r}$, $\alpha$ is the coefficient of migration delay, and $h_{r,\hat{r}}$ is the number of hops between $r$ and $\hat{r}$.

The processing delay for the task $i$ is calculated as:

$$T^p_{r,\hat{r},i}(t) = \frac{x_{r,i}(t) \cdot f_i}{F_{r,i}} + \frac{x_{r,\hat{r},i}(t) \cdot f_i}{F_{\hat{r},i}}, \quad \forall i \in \mathcal{J}, \quad \forall r, \hat{r} \in \mathcal{R} \cup \{\text{BS}\} \setminus \{r\}.$$  (4)

where $f_i$ is the number of CPU cycles required to completely process the task $i$, and $F_{r,i}$ and $F_{\hat{r},i}$ cycles/s are respectively the computation capacity allocated to the task $i$ by $r$ and $\hat{r}$.
From Eqs. (2), (3) and (4), the total delay to complete the task \( i \) is calculated as:
\[
T_{r,\hat{r},i}(t) = T_{r,i}(t) + T_{r,\hat{r},i}^m(t) + T_{r,\hat{r},i}^p(t).
\] (5)

Similar to [5], the cost for processing the task \( i \) is given by:
\[
c_{r,\hat{r},i}(t) = c_{r,\hat{r},i}^a(t) + c_{r,\hat{r},i}^m(t) + c_{r,\hat{r},i}^p(t),
\]
where \( c_{r,\hat{r},i}^a(t) \) and \( c_{r,\hat{r},i}^m(t) \) are the costs of task uploading, cost migrating and task processing.

Specifically,
\[
c_{r,\hat{r},i}^m(t) = \delta_m^r \cdot B_{r,i}(t), \quad \forall r \in \mathcal{R}(\{\hat{r}\}),
\] (7)
where \( \delta_m^r > 0 \) is the migration cost and \( \theta \) is the data size of each service entity.

The computation cost for the task \( i \) at either RSU/BS \( r \) or \( \hat{r} \) is expressed as:
\[
c_{r,\hat{r},i}^p = x_{r,\hat{r},i}(t) \cdot \delta_p^r \cdot F_{r,i} + x_{r,\hat{r},i}(t) \cdot \delta_p^{\hat{r}} \cdot F_{\hat{r},i},
\]
where \( \delta_p^r > 0 \) and \( \delta_p^{\hat{r}} > 0 \) are the unit computation costs.

### D. Problem Formulation

There are three constraints that describe the dependence of task assignment on the vehicle mobility. When a task \( i \) is completed by an RSU \( r \) or \( \hat{r} \), the delay for completing the task \( i \) must not be larger than the duration that the vehicle \( i \) resides within \( r \)'s coverage area. As such,
\[
x_{r,i}(t) \left[ T_{r,i}(t) - \frac{d_{r,i}(t)}{v_i} \right] \leq 0, \quad \forall r \in \mathcal{R} \setminus \{\hat{r}\},
\] (10)
where \( d_{r,i}(t) \) is the distance that the vehicle \( i \) travels before leaving the coverage area of \( r \).

If the task \( i \) is migrated from the RSU \( r \) to another RSU \( \hat{r} \), the delay is instead constrained by:
\[
x_{r,\hat{r},i}(t) \left[ T_{r,\hat{r},i}(t) - \frac{d_{r,\hat{r},i}(t) + D_R \cdot h_{r,\hat{r}}}{v_i} \right] \leq 0, \quad \forall r \in \mathcal{R} \setminus \{\hat{r}\},
\] (11)

If the task \( i \) is migrated from the BS to an RSU \( \hat{r} \), the delay is then constrained by:
\[
x_{r,\hat{r},i}(t) \left[ T_{r,\hat{r},i}(t) - \frac{d_{r,\hat{r},i}(t) + 2R_R + D_R \cdot h_{r,\hat{r}}}{v_i} \right] \leq 0, \quad \forall i \in \mathcal{J}, r \in \{\{\text{BS}\}, \forall r \in \mathcal{R} \setminus \{\hat{r}\},
\] (12)
where \( d_{r,\hat{r},i} \) is the distance that the vehicle \( i \) has traveled in the area uncovered by any RSUs before it enters the coverage area of the closest RSU \( \hat{r} \), and \( R_R \) is the communication range of an RSU.

We aim to minimize the total delay and cost for completing all \( |\mathcal{J}| \) tasks. The task assignment is thus formulated as the following BNLP problem.

\[
\begin{align*}
\min_{x_{r,i}(t)} & \sum_{i \in \mathcal{J}} \sum_{r \in \{\{\text{RSUs}\}, \forall r \in \mathcal{R} \setminus \{\hat{r}\}\}} \left[ \beta : T_{r,\hat{r},i}(t) + \gamma \cdot c_{r,\hat{r},i}(t) \right] \\
\text{s.t.} & \sum_{i \in \mathcal{J}} x_{r,i}(t) \cdot F_{r,i} \leq F_{r}^{\text{max}}, \quad \forall r \in \mathcal{R} \\
& \sum_{i \in \mathcal{J}} x_{r,i}(t) \cdot F_{\hat{r},i} \leq F_{\hat{r}}^{\text{max}}, \quad \forall r \in \mathcal{R} \\
& \sum_{i \in \mathcal{J}} x_{r,\hat{r},i}(t) \cdot F_{r,\hat{r},i} \leq F_{r,\hat{r}}^{\text{max}}, \quad \forall r \in \mathcal{R} \\
& \forall r \in \mathcal{R} \cup \{\text{BS}\}, \forall r \in \mathcal{R} \cup \{\hat{r}\},
\end{align*}
\] (13)
where \( \beta > 0 \) and \( \gamma > 0 \) are the weights to prioritize the delay and the cost, respectively. Constraints (13e), (13f) and (13h) show that the computation capacity assigned to a task \( i \) is upper-bounded by the maximum computation capacities \( F_{r}^{\text{max}} \) or \( F_{\hat{r}}^{\text{max}} \).
Constraint (13i) enforces that an arbitrary task \( i \) must be processed by one of the RSUs and the BS.

### III. PROPOSED REGRET-MATCHING LEARNING BASED TASK ASSIGNMENT SCHEME

The optimization problem in (13) is nonconvex and combinatorial with nonlinear constraints. Traditional optimization methods may not be able to return a solution within an acceptable time frame, which is an important requirement in vehicular networks with a high degree of mobility. As such, we propose an iterative game-based learning algorithm that guarantees an equilibrium solution. The proposed algorithm is based on the regret minimization procedure [9], [10]. This procedure is well-known for its low complexity and provable convergence guarantee when making decision in a situation with multiple stakeholders. We further introduce a forgetting factor in the learning algorithm to enable fast adaptation of the learning agents (vehicles) to changes in the environment when they move. Using simulations with realistic network settings, we will later show that our solution adapts and converges much faster than existing approaches, especially when the number of participating vehicles (i.e., tasks) increases.

#### A. Game Reformulation for Task Assignment [13]

We propose to reformulate problem (13) as a multi-agent distributed learning problem. Here, each requesting vehicle is an independent decision maker who learns to jointly achieve
a globally optimal solution. Specifically, we model the task assignment in VEC as a repeated game $G = (J, A, U)$, where players aim to minimize the long-run average delay and cost of processing the tasks of the requesting vehicles.

In this model, the (finite) set of requesting vehicles $J = \{1, 2, \ldots, |J|\}$ is regarded as the set of players. Each player $i \in J$ has its set of finite actions $A_i = \mathbb{R} \cup \{\text{BS}\}$ as it decides where to offload its task to. We denote by $\mathcal{A} = A_1 \times A_2 \times \cdots \times A_{|J|}$ the set of joint actions of all players. Let $U = \{u_1, u_2, \ldots, u_{|J|}\}$ denote the set of utility functions of all the players.

To minimize the delay and cost for processing the tasks for the vehicles, the utility function of a player $i \in J$ at a time $t$ resulting from a given action $a_i^{(t)} = r \in A_i$ is designed as:

$$u_i^{(t)}(a_i^{(t)}, a_{-i}^{(t)}) = -\left[\beta \cdot T_{a_i^{(t)}, a_{-i}^{(t)}}(t) + \gamma \cdot c_{a_i^{(t)}, a_{-i}^{(t)}}(t)\right],$$

(14)

where $a_{-i}^{(t)}$ denotes the vector of RSU/BS actions decided by all the other $|J| - 1$ players at the time $t$. The parameters $T_{a_i^{(t)}, a_{-i}^{(t)}}(t)$ and $c_{a_i^{(t)}, a_{-i}^{(t)}}(t)$ are calculated by (5) and (6), respectively. Under this utility model, each player $i$ obtains a player-specific payoff depending on the joint action profile $(a_1^{(t)}, \ldots, a_{|J|}^{(t)})$ over all players. Here, maximizing the sum of all the players’ utilities would result in minimizing the objective function in problem (13).

B. Correlated Equilibrium

In most cases, a game-based solution guarantees convergence to a set of equilibria, in which any vehicle does not achieve any gain by unilaterally changing their decision. It can be shown that the equilibrium of the reformulated game $\bar{G}$ is a correlated equilibrium (CE) \cite{11, 12}. A probability distribution $\psi$ defined on $\mathcal{A}$ is said to be a CE if for all player $i \in J$, for all $a_{-i} \in A_{-i}$, and for every pair of action $j, k \in A_i$, it holds true that

$$\sum_{a_{-i} \in A_{-i}} \psi(j, a_{-i}) u_i(k, a_{-i}) - u_i(j, a_{-i}) \leq 0.$$  

(15)

When in a CE each player does not benefit from choosing any other probability distribution over its actions, provided that all the other players do likewise.

C. Regret-based Learning with a Forgetting Factor

An iterative algorithm that can be used to reach the CE set is the regret matching procedure proposed in \cite{10}. The key idea is to adjust the player’s action probability to be proportional to the “regrets” for not having played other actions. Specifically, for any two actions $j \neq k \in A_i$ at any time $t$, the cumulative regret of a player $i$ up to a time $t$ for not playing action $k$ instead of its actually played action $a_i^{(t)} = j$ is

$$D_i^{(t)}(j, k) = \frac{1}{t} \sum_{s=1}^t \mathbb{I}\{a_i^{(s)} = j\} \left[u_i^{(s)}(k, \cdot) - u_i^{(s)}(j, \cdot)\right],$$

where $\mathbb{I}(.)$ denotes the indicator function. This is the change in the average payoff that the player $i$ would have if choosing a different action $k \neq j$ every time they played $j$ in the past, given that all other players did not change their decisions. A positive value indicates a “regret” by the player $i$ for not having played action $k$ instead of the chosen action $j$.

The regret can be recursively expressed as:

$$D_i^{(t)}(j, k) = \left(1 - \frac{1}{t}\right) D_i^{(t-1)}(j, k) + \frac{1}{t} D_i^{(t)}(j, k),$$

(16)

where $D_i^{(t)}(j, k) = \mathbb{I}\{a_i^{(t)} = j\} \left[u_i^{(t)}(k, a_{-i}) - u_i^{(t)}(j, a_{-i})\right]$ denotes the instantaneous regret by the player $i$ for not playing the action $k$ instead of its played action $j$ at the time $t$. Equation (16) updates the cumulative regret at each time step by adding a correction term based on the new instantaneous regret. As a stochastic approximation method, \cite{16}, although resulting in almost surely convergence, can be quite slow. This is especially true in a dynamic environment, where player’s utility changes with time. It is likely to become a major issue in our considered vehicular networking scenario with a high degree of mobility.

To this end, we will now introduce a forgetting factor for updating $D_i^{(t)}(j, k)$ as:

$$\bar{D}_i^{(t)}(j, k) = \lambda \bar{D}_i^{(t-1)}(j, k) + (1 - \lambda) D_i^{(t)}(j, k),$$

(17)

where $0 \leq \lambda \leq 1$ is a forgetting factor used to regulate the influence of outdated values of regret over the instantaneous regret. Each player then independently chooses its next action according to the following probabilities:

$$\pi_i^{(t+1)}(k) = \frac{1}{|A_i|} \left|\bar{D}_i^{(t)}(j, k)\right|^+, \quad (18)$$

$^1|x|^+ = \max\{x, 0\}$ for a real number $x$.

---

**Algorithm 1 RM Learning-Based Task Assignment Algorithm**

1: **Initialization**: Each player $i$ initializes its action selection policy with a uniform strategy $\pi_i^{(1)}(j) = \frac{1}{|A_i|}$ for all $j \in A_i$.
2: **Main algorithm**: Each player $i \in J$ independently runs the following procedure to decide its action over time
3: for $t = 1, 2, \ldots$ do
4: **Action selection**: Player $i$ samples an action $a_i^{(t)} = j \in A_i$ from its probability distribution of action selection $\pi_i^{(t)}(j)$, and updates its chosen action to all other players.
5: **Utility update**: Player $i$ receives a utility as a result of its chosen action $a_i^{(t)}(j, a_{-i}^{(t)})$ computed by Eq. (14).
6: for $k \in A_i \setminus \{j\}$
7: **Expected utilities**: Using Eq. (14), player $i$ calculates an expected utility $u_i^{(t)}(k, a_{-i}^{(t)})$ if choosing an action $k \neq j$, given the choices made by the other players.
8: **Regret update**: Using Eq. (17), player $i$ computes the cumulative regret $\bar{D}_i^{(t)}(j, k)$ for not choosing $k$.
9: **Strategy update**: Using Eq. (18), player $i$ updates its next action probability $\pi_i^{(t+1)}(k)$. 
10: end for
11: Player $i$ plays the same action chosen in the previous round with the remaining probability $\pi_i^{(t+1)}(j) = 1 - \sum_{k \neq j} \pi_i^{(t+1)}(k)$.
12: end for

---
for all \( k \neq j \), and \( \mu \) is chosen such that the probability of playing the same action in the next iteration is positive. The pseudo-code of our distributed algorithm, which is run independently by each agent, is shown in Algorithm 1.

We now state the main theoretical result of our proposed game-based solution to the original problem (13). Note that only a sketch of the proof is provided due to space constraint.

**Theorem 1.** If every player chooses their actions according to (18), then the joint empirical distribution of action profiles converges almost surely to the CE set of the game \( S \) as \( t \to \infty \).

**Proof:** For notational convenience, let us drop the subscript \( t \) and define the following Lyapunov function:

\[
P(\bar{D}) = \frac{1}{2} \left( \text{dist}[\bar{D}, \mathbb{R}^-] \right)^2 = \frac{1}{2} \sum_{j,k} (|\bar{D}(j,k)|^+) \, ,
\]

(19) where \( \mathbb{R}^- \) represents the negative orthant. Taking the time-derivative of (19) yields

\[
\frac{d}{dt} P(\bar{D}) = \sum_{j,k} |\bar{D}(j,k)|^+ \cdot \frac{d}{dt} \bar{D}(j,k) \, .
\]

(20) First, we find \( \frac{d}{dt} \bar{D}(j,k) \) by rewriting (17) as:

\[
\bar{D}^{(t)}(j,k) = \bar{D}^{(t-1)}(j,k) + (1-\lambda) \left\{ \bar{D}^{(t-1)}(j,k) - \bar{D}^{(t-1)}(j,k) \right\}
\]

(21) Let \( \epsilon = 1 - \lambda \) be a constant step size. It can be seen that (21) has the form of a constant step size stochastic approximation algorithm \( \theta_{k+1} = \theta_k + \epsilon \hat{H}(\theta_k, x_k) \) and satisfies (13) Th. 17.1.1]. Thus, its dynamics can be characterised by an ordinary differential equation (see (13) Ch. 17) for further details). This means the system can be approximated by replacing \( x_k \) with its expected value. By applying (13) Theorem 17.1.1], \( \bar{R}_t(j,k) \) converges weakly (in distribution) to the averaged system corresponding to (21). As such, we find

\[
\frac{d}{dt} \bar{D}(j,k) = E \left\{ [u(k, \cdot) - u(j, \cdot)] \mathbb{I}[a_i^{(t)} = j] - \bar{D}(j,k) \right\}
\]

(22) Next, replacing \( \frac{d}{dt} \bar{D}(j,k) \) from (22) into (20) gives:

\[
\frac{d}{dt} P(\bar{D}) = \sum_{j,k} |\bar{D}(j,k)|^+ \left( U(k, \ell) - U(j, \ell) \right) \, \pi(j)
\]

\[
- \sum_{j,k} |\bar{D}(j,k)|^+ \times \bar{D}(j,k)
\]

\[
\leq \frac{2G\delta}{|A_i|} \sum_{j,k} |\bar{D}(j,k)|^+ - 2P(\bar{D})
\]

(23) where \( G \) is an upper bound on \( |u(\cdot)| \), \( 0 \leq \delta \leq 1 \), and \( |A_i| \) is the cardinality of the action set \( A_i \) (the set of actions of a player \( i \)). Note that in the last equality of (23), we have used the following two lemmas:

\[
\sum_{j,k} |\bar{D}(j,k)|^+ \leq 2P(\bar{D}) \text{ (immediate from Eq. (19))}
\]

(2) \( \sum_{j,k} |\bar{D}(j,k)|^+ u(k, \cdot) - u(j, \cdot) \pi(j) \leq \frac{2G\delta}{|A_i|} \sum_{j,k} |\bar{D}(j,k)|^+ \).

Finally, it follows that by assuming \( |\bar{D}(j,k)|^+ \geq \kappa > 0 \), one can choose a sufficiently small value of \( \delta > 0 \) such that

\[
\frac{d}{dt} P(\bar{D}) \leq -\delta P(\bar{D})
\]

This implies that \( P(\bar{D}(t)) \) goes to zero at an exponential rate. Therefore, \( \lim_{t \to \infty} \text{dist}[\bar{D}_t, \mathbb{R}^-] = 0 \).

Let \( \phi_t \) be the empirical distribution of the joint action \( (j, a_i^{(t)}) \) by all players up to the time \( t \). It can be defined by a stochastic approximation recursion as:

\[
\phi_t (j, a_i^{(t)}) = \phi_{t-1} (j, a_i^{(t-1)})
\]

\[
+ \epsilon \left\{ a_i^{(t)} \right\} \mathbb{P}[a_i^{(t)} = j] - \phi_{t-1} (j, a_i^{(t-1)})
\]

\[
= \epsilon \sum_{\tau < t} (1-\epsilon)^{t-\tau} \mathbb{P}[a_i^{(t)} = j]
\]

(24) The elements of the regret matrix in (16) can be rewritten as follows:

\[
\hat{D}_{i,j}(k,i) = \epsilon \sum_{\tau < t} (1-\epsilon)^{t-\tau} \left( u_{i,k} - u_{i,j} \right) \mathbb{P}[a_i^{(t)} = j]
\]

\[
= \epsilon \sum_{\tau < t} (1-\epsilon)^{t-\tau} \mathbb{P}[a_i^{(t)} = j] \mathbb{P}[a_i^{(t)} = j]
\]

\[
= \epsilon \sum_{\tau < t} (1-\epsilon)^{t-\tau} \mathbb{P}[a_i^{(t)} = j] \left( u_{i,k} - u_{i,j} \right)
\]

\[
= \epsilon \sum_{\tau < t} (1-\epsilon)^{t-\tau} \mathbb{P}[a_i^{(t)} = j] \left( u_{i,k} - u_{i,j} \right)
\]

(25) On the last line of (25), we have substituted \( \phi_t (j, a_i^{(t)}) \) from (24). Finally, on any convergent subsequence \( \lim_{t \to \infty} \phi_t \to \psi \), we have:

\[
\lim_{t \to \infty} \hat{D}_{i,j}(k,i) = \sum_{\tau < t} (1-\epsilon)^{t-\tau} \left( u_{i,k} - u_{i,j} \right)
\]

(26) Comparing (25) with the definition of CE in Eq. (15) completes the proof.

**IV. PERFORMANCE EVALUATION**

**A. Simulation Settings**

We evaluate the performance of our proposed Algorithm 1 through numerical experiments in MATLAB (ver. 2020b). We consider a six-lane highway with three lanes in each direction, as depicted in Fig. 1. Similar to (1), (2), (6), we set \( |R| = 10 \), \( |S| = [25; 100] \), \( B_H = [3; 6] \text{ km} \), \( B_R = 600 \text{ m/s} \), \( s_i \in [200; 500] \text{ MB} \), \( \theta = 500 \text{ MB} \), \( f_i \in [0.5; 1.2] \text{ Gcycles} \), \( B_R = 100 \text{ MHz} \), \( \alpha = 0.02 \text{ s/hop} \), \( \delta_{r,m} = 0.002 \text{ unit/MHz} \), \( p_i = 20 \text{ dBm} \). If \( r \in \{B_S\} \), then we set \( B_{r,i} = 0.25 \text{ MHz} \), \( \delta_{r} = 20 \text{ units/MHz} \), \( f_{i,\text{max}} = 20 \text{ GHz} \), \( F_{r,i} \in [1; 20] \text{ GHz} \), and \( \delta_{r} = 100 \text{ units/GHz} \). Otherwise, we set \( B_{r,i} = 1 \text{ MHz} \), \( \delta_{r} = 2 \text{ units/MHz} \), \( f_{i,\text{max}} = 10 \text{ GHz} \), \( F_{r,i} \in [1; 10] \text{ GHz} \) and \( \delta_{r} = 10 \text{ units/GHz} \). We set \( \beta = \gamma = 1 \) and \( \lambda = 0.2 \). The vehicle speeds in lanes \{1, 4\}, \{2, 5\} and \{3, 6\} are set as 90,
100 and 120 km/h, respectively [15]. The time step \( t \) for one iteration is set as 1 s.

For comparison, we benchmark Algorithm 1 against the three following relevant schemes: (1) Without Migration scheme (WM), (2) Reinforcement Learning with Network-Assisted Feedback scheme (RLNF) [7], and (3) Traditional Regret-Matching scheme (TRM) [9].

B. Simulation Results

To compare the convergence performance of the four algorithms, Fig. 2 depicts the total delay and cost of completing 50 tasks. It is clear that Algorithm 1 converges to the best solution here—the CE solution. Although the TRM and RLNF schemes are also based on RM learning, they are not able to perform as well. That is because the regret values in TRM and RLNF are not updated with respect to changes in the operating vehicular environment. Furthermore, because TRM makes task assignment decisions based on information about global network conditions, its total delay and cost is lower than that of RLNF. In the WM scheme, the constraints of computing capacity and completion delay in problem (13) are violated because task migration does not happen. As seen from Fig. 2, WM is the worst performer.

Fig. 3 shows that as the number of participating vehicles (equivalently, the number of tasks to be processed) grows, Algorithm 1 achieves an even much faster convergence speed than does the second-best TRM algorithm. This important result demonstrates the advantage of our proposal in large-scale problems, which are common in vehicular networks.

Fig. 4 shows that as the inter-RSU distance decreases, the objective value of problem (15) also decreases. This result confirms that one must increase the density of RSU deployment (i.e., decrease the inter-RSU distance) to reduce total delay and cost of task offloading. In that case, the vehicles would have more chance to completely process their tasks at their serving RSUs, resulting in a shorter delay and a lower cost. In any case, Algorithm 1 achieves the best performance.

V. Conclusion

To address the issue of limited computation resources in VEC, this paper proposes a task assignment scheme where vehicles’ tasks are migrated across multiple servers according to their mobility pattern. To this end, we have formulated a BNLP problem that minimizes the total delay and cost incurred by the participating vehicles. We have then proposed an RM learning-based algorithm that is theoretically proved to converge to the CE solution to the formulated problem. Simulation results show the clear advantages of our proposed algorithm over existing solutions.

REFERENCES

[1] L. T. Tan and R. Q. Hu, “Mobility–Aware Edge Caching and Computing in Vehicle Networks: A Deep Reinforcement Learning,” IEEE Trans. Veh. Technol., vol. 67, no. 11, pp. 10 190–10 203, Nov. 2018.
[2] Z. Ning, K. Zhang, X. Wang, M. S. Obaidat, L. Guo, X. Hu, B. Hu, Y. Guo, B. Sadoun, and R. Y. K. Kwok, “Joint Computing and Caching in 5G-Envisioned Internet of Vehicles: A Deep Reinforcement Learning-Based Traffic Control System,” IEEE Trans. Intell. Transp. Syst., pp. 1–12, 2020.
[3] W. Zhan, C. Luo, J. Wang, C. Wang, G. Min, H. Duan, and Q. Zhu, “Deep-Reinforcement-Learning-Based Offloading Scheduling for Vehicular Edge Computing,” IEEE Internet Things J., vol. 7, no. 6, pp. 5449–5465, 2020.
[4] H. Peng and X. S. Shen, “Deep Reinforcement Learning Based Resource Management for Multi-Access Edge Computing in Vehicular Networks,” IEEE Trans. Netw. Sci. Eng., pp. 1–1, 2020.
[5] H. Wang, H. Ke, G. Liu, and W. Sun, “Computation Migration and Resource Allocation in Heterogeneous Vehicular Networks: A Deep Reinforcement Learning Approach,” IEEE Access, vol. 8, pp. 171 140–171 153, 2020.
[6] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, “A Joint Service Migration and Mobility Optimization Approach for Vehicular Edge Computing,” IEEE Trans. Veh. Technol., vol. 69, no. 8, pp. 9041–9052, 2020.
[7] D. D. Nguyen, H. X. Nguyen, and L. B. White, “Reinforcement Learning With Network-Assisted Feedback for Heterogeneous RAT Selection,” IEEE Trans. Wireless Commun., vol. 16, no. 9, pp. 6062–6076, 2017.
[8] C. Fan, B. Li, C. Zhao, and Y. Liang, “Regret Matching Learning Based Spectrum Reuse in Small Cell Networks,” IEEE Trans. Veh. Technol., vol. 69, no. 1, pp. 1060–1064, 2020.
[9] S. Hart and A. Mas-Colell, “A Simple Adaptive Procedure Leading to Correlated Equilibrium,” Econometrica, vol. 68, no. 5, pp. 1127–1150, 2000.
[10] ——, “A Reinforcement Procedure Leading to Correlated Equilibrium,” in Economics Essays. Berlin: Springer, 2001, pp. 181–200.
[11] R. J. Aumann, “Correlated Equilibrium as an Expression of Bayesian Rationality,” Econometrica: Journal of the Econometric Society, vol. 55, no. 1, pp. 1–18, Jan 1987.
[12] J. Hart and D. Schmeidler, “Existence of Correlated Equilibria,” Mathematics of Operations Research, vol. 14, no. 1, pp. 18–25, Feb 1989.
[13] V. Krishnamurthy, Partially Observed Markov Decision Processes From Filtering to Controlled Sensing. Cambridge University Press, 2016.
[14] D. D. Nguyen, “Adaptive Reinforcement Learning for Heterogeneous Network Selection,” Ph.D. dissertation, The University of Adelaide, 2018.
[15] B. L. Nguyen, D. T. Ngo, M. N. Dao, Q. T. Duong, and M. Okada, “Joint Scheduling and Power Control Scheme for Hybrid 2V/2V Networks,” IEEE Trans. Veh. Technol., vol. 69, no. 12, pp. 15 668–15 681, 2020.