Multi-Agent Task Assignment in Vehicular Edge Computing: A Regret-Matching Learning-Based Approach

Bach Long Nguyen, Duong D. Nguyen, Hung X. Nguyen, Duy T. Ngo, Senior Member, IEEE, and Markus Wagner

Abstract—Vehicular edge computing has emerged as a solution for enabling computation-intensive applications within Intelligent Transportation Systems (ITS), encompassing domains like autonomous driving and augmented reality. Despite notable progress in this domain, the efficient allocation of constrained computational resources to a spectrum of time-critical ITS tasks remains a substantial challenge. We address this challenge by devising an innovative task assignment scheme tailored for vehicles navigating a highway. Given the high speed of vehicles and the limited communication radius of roadside units (RSUs), the dynamic migration of computation tasks among multiple servers becomes imperative. We present a novel approach that formulates the task assignment challenge as a binary nonlinear programming (BNLP) problem, managing the allocation of computation tasks from vehicles to RSUs and a macrocell base station. To tackle the potentially large dimensionality of this optimization problem, we develop a distributed multi-agent regret-matching learning algorithm. Incorporating the method of regret minimization, our proposed algorithm employs a forgetting mechanism that enables a continuous learning process, thereby accommodating the high mobility of vehicle networks. We prove that this algorithm converges towards correlated equilibrium solutions for our BNLP formulation. Extensive simulations, grounded in practical parameter settings, underscore the algorithm’s ability to minimize total delay and task processing costs, while ensuring equitable utility distribution among agents.

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

I. INTRODUCTION

Given the constrained computational and storage capabilities inherent to vehicular users, fulfilling the stringent demands of Intelligent Transportation System (ITS) applications, characterized by computationally intensive tasks and content caching with minimal latency, presents a major challenge [1], [2]. In response, Vehicular Edge Computing (VEC), as an application of Mobile Edge Computing (MEC) tailored for high-mobility scenarios, has been proposed as a solution [3], [4], [5], [6]. Even so, there remains the 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.

We consider network scenarios as depicted in Fig. 1. Specifically, autonomous vehicles left-drive in two directions on a six-lane highway, similar to M1 Pacific Highway linking Newcastle with Sydney in New South Wales, Australia [7]. A macrocell base station (BS) is deployed to provide network connectivity along the highway. To facilitate data and computation offloading, a set $\mathcal{R}$ of road side units (RSUs) are deployed at an inter-RSU distance of $D_R$, enhancing network proximity to 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.

Let us assume that the vehicles have to complete computation-intensive tasks. Given their constrained computational capabilities, 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. We 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 $j \in \mathcal{J}$ 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 $v_i$, it 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 $\mathcal{R}$.
A. Background

To address the challenge of insufficient allocation of computation resources for numerous users, [8] proposes an algorithm that optimally distributes tasks from smart mobile devices (SMDs) to MEC servers. By using a combination of particle swarm optimization, simulated annealing and genetic algorithms, this approach minimizes the energy consumed by SMDs and servers while also optimizing the task offloading ratio. In a related work by [9], tasks sent from the computers and iPads in the terminal layer are allocated to servers in the edge computing layer and cloud data layer. With the aim of maximizing the overall network profit (which is the net revenue discounted by a cost), the authors propose a task allocation strategy that utilises a swarm intelligence approach based on simulated annealing. However, without taking the mobility of SMD users into account in the problem formulation, these two algorithms are only applicable to static environments, rather than high-mobility environments (i.e. in ITS or vehicular networks).

Unlike [8], [9], the work conducted by [10] devises a task assignment algorithm where tasks requested by vehicles are assigned to either VEC servers or volunteer vehicles in a vehicular network. The developed algorithm is based on a Stackelberg game where VEC servers and requesting vehicles are respectively modelled as leaders and followers. To completely process all the tasks, the servers recruit more volunteers while setting and sending prices to the requesting vehicles. The game strategy is to 1) maximize the income of VEC servers and volunteer vehicles, 2) reduce the cost incurred to the servers and volunteers, and 3) minimize the payment made by the requesting vehicles for processing their tasks. However, when the requesting vehicles and volunteer vehicles move in different directions and at different speeds, their connection time is limited to a brief amount due to the short communication range of vehicles (about 300 m). Consequently, the requesting vehicles will be out-of-range of the volunteer vehicles, while the latter have not completely processed the tasks requested by the former.

The optimization methods for resource allocation proposed in [8], [9], [10], [11], [12] require accurate knowledge of channel conditions. These conditions are inherently dynamic, and oftentimes unavailable in high-mobility scenarios. The solutions are typically based on a snapshot model of the vehicular networks, while ignoring the long-term influence of the current decision [3]. By contrast, without any prior knowledge of the operating environment, reinforcement learning (RL) is able to make decisions that maximize the long-term rewards for the networks according to [13] and [14]. 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 and unknown channel conditions.

In [15], 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. To minimize the delay of completing vehicles’ tasks, [16] proposes a cross-entropy-based learning algorithm for sharing limited frequency channels among the vehicles. Meanwhile, [17] relies on deep reinforcement learning (DRL) for optimizing jointly transmit power, frequency channel selection and computation resource allocation. In the two cases, tasks are however not migrated among servers. Taking a step further, a joint management scheme of spectrum, computing and storing resources in VEC is proposed in [18] using DRL. Note that in [15], [16], [17], [18], 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 (about 600 m); hence, it is possible that a vehicle moves out of the range of its serving RSU even before that RSU processes its tasks completely.

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. For example, in [19], there is an autonomous vehicle moving along a highway or a city expressway, and its tasks are migrated between MEC servers and

---

Fig. 1. Task assignment in VEC-based ITS applications.
processed. Assuming these MEC servers have large computation resources, [19] use DRL to minimize the energy consumption for task processing while meeting latency requirements. In addition, only a single agent interacts with the environment to determine an optimal task migration policy. Also based on DRL, [20] not only develop a task migration scheme but also find the best migration routes for vehicles in urban areas. 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. Compared to [19], the work of [20] could be applicable to multi-agent systems owing to utilizing communication and cooperation between multiple autonomous vehicles. However, this scheme might cause a change in the original route of vehicles as the tasks are not migrated with respect to the vehicles’ mobility pattern.

In practice, one major difficulty in solving multi-agent systems as in [20] lies in the uncertainty of the algorithm convergence, particularly when applied in large environments involving a substantial number of agents, e.g., 100 or more. To address this issue, [21] and [22] employ regret matching (RM) learning to design algorithms for multi-agent systems. The advantages of RM learning-based algorithms in several applications, e.g., seasonal forecasting and learning in matching markets without incentives, have been demonstrated by [23], [24], [25], [26]. In particular, these algorithms can converge to correlated-equilibrium solutions faster than RL-based algorithms as shown in [27] and [28]. Additionally, it is unnecessary that the correlated equilibrium solutions must be the optimal solution. Their algorithms, however, were not specifically designed for task migration in VEC, and their solutions may be rendered inapplicable due to the inherited characteristics of vehicle networks with high mobility.

B. Contributions

In this paper, we introduce a task assignment scheme building upon regret matching learning. The primary objective is to minimize the total delay and cost incurred by vehicles in highway scenarios like [19]. We assume that as soon as a vehicle exits the coverage area of its designated RSU, it will proceed to transfer 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) To improve over [15] and [18], we formulate a task assignment problem as a binary nonlinear programming (BNLP) problem with specific constraints on the movement of participating vehicles. Compared to [19] and [20], this problem is formulated for migrating the tasks of multiple autonomous vehicles between servers according to these vehicles’ movement.

2) Unlike [19] and [20], 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. In particular, this iterative game-based learning algorithm is able to guarantee an equilibrium solution. We further propose a forgetting method to speed up the convergence of the traditional RM algorithms in [21], [22]. Doing so allows the algorithm to effectively handle the high level of user mobility in vehicle networks.

3) Our simulation results with practical parameter settings showcase the strengths of our proposed solution in terms of delay and cost minimization, utility fairness among agents, and convergence speed particularly in large-scale network settings.

The subsequent sections of this paper is organized as follows. In Section II, we present the system model, including both a wireless communication model and a computation model. Section III describes the problem formulation for task assignment. Then, Section IV introduces the RM-based solution to the task assignment challenge. Here, the problem is reformulated as a repeated game while the definition of a correlated equilibrium is introduced. To illustrate the efficacy of our proposed methodology, Section V conducts simulations. Finally, we summarize the paper in Section VI.

II. SYSTEM MODEL

In our scenarios, once a requesting vehicle wants its task to be processed by a server at an RSU or a BS, it must send the task to the RSU/BS via a wireless link. In addition, the task can be migrated from the RSU/BS to the others via a wired connection. Thus, we first model wireless communication between the requesting vehicles and the RSUs/BS in Section II-A. Then, to determine the amount of time and cost needed for 1) uploading tasks through wireless links, 2) migrating tasks between RSUs/BS through wired links, and 3) processing tasks completely, we develop a computation model in Section II-B. The delay and cost will be used for our problem formulation in Section III. In particular, important notations are summarized in Table I.

A. 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 $i \in \mathcal{I}$. 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:

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

(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. The link gain $|h_{r,i}(t)|^2$ is time-varying with respect to the bandwidth $B_{r,i}(t)$. 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 $t$. In addition, like [19], [20], the received noise power $N_r^2$ is assumed to be static.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE I

| Notations | Descriptions |
|-----------|--------------|
| $r \in \mathbb{R} \cup \{BS\}$ | Set of servers that can process vehicles’ tasks |
| $i \in \mathcal{J}$ | Set of tasks requested by vehicles |
| $T_{r,i}^u(t), T_{r,i}^m(t), T_{r,i}^p(t)$ | Delay for uploading task $i$ from vehicle $i$ to server $r$, migrating task $i$ between servers $r$ and $\hat{r}$, and processing task $i$, respectively |
| $T_{r,\hat{r},i}(t)$ | Total delay for completing a task $i$ |
| $c_{r,t,i}(t), c_{r,\hat{r},i}(t)$ | Cost of uploading task $i$ from vehicle $i$ to server $r$, migrating task $i$ between servers $r$ and $\hat{r}$, and processing task $i$, respectively |
| $c_{r,\hat{r},i}(t)$ | Total cost of completing task $i$ |
| $x_{r,i}(t), x_{r,\hat{r},i}(t)$ | Decision variables indicating task $i$ is processed by server $r$, or migrated from server $r$ to server $\hat{r}$ then processed by server $\hat{r}$, respectively |
| $A_r, A_\hat{r}$ | Action set of vehicle $i$ and the remaining $|\mathcal{J}\setminus\{i\}|$ vehicles, respectively |
| $a_r(t), a_\hat{r}(t)$ | Actions played by vehicles $i$ and the remaining $|\mathcal{J}\setminus\{i\}|$ vehicles, respectively, at time $t$ |
| $u_{r,i}(t), a_{r,i}(t)$ | Utility of vehicle $i$ at time $t$ when vehicle $i$ and the remaining $|\mathcal{J}\setminus\{i\}|$ vehicles perform actions $a_r(t)$ and $a_{r,i}(t)$, respectively |
| $D_r(t)(j,k), D_{\hat{r}}(t)(j,k)$ | Cumulative regret and instantaneous regret of vehicle $i$ when it plays action $j$ instead action $k$ at time $t$ |
| $\pi_{r,i}^{(t+1)}(k)$ | Probability that vehicle $i$ plays action $k$ at time $t+1$ |

B. Computation Model

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

$$
T_{r,i}^u(t) = \frac{s_i}{R_{r,i}(t)}, \quad \forall r \in \mathbb{R} \cup \{BS\},
$$

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 $r \in \mathbb{R} \cup \{BS\}$, 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 $\hat{r} \in \mathbb{R} \cup \{BS\} \setminus \{r\}$, then $x_{r,\hat{r},i}(t) = 1$; otherwise, $x_{r,\hat{r},i}(t) = 0$. By building on [20], the task migration time is calculated as:

$$
T_{r,\hat{r},i}^m(t) = x_{r,\hat{r},i}(t) \left( \frac{s_i}{B_R} + 2 \cdot \alpha \cdot h_{r,\hat{r}} \right), \quad \forall r \in \mathbb{R} \cup \{BS\}, \quad \forall \hat{r} \in \mathbb{R} \cup \{BS\} \setminus \{r\},
$$

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 $\hat{r}$. In addition, like [20], we define by $2 \cdot \alpha$ the average delay of round-trip communication between two consecutive servers when a task is migrated by the server to the other one.

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

$$
T_{r,\hat{r},i}^p(t) = \frac{x_{r,i}(t) \cdot f_i + x_{r,\hat{r},i}(t) \cdot f_{\hat{r}}}{F_{r,i}}, \quad \forall r \in \mathbb{R} \cup \{BS\}, \quad \forall \hat{r} \in \mathbb{R} \cup \{BS\} \setminus \{r\},
$$

where $f_i$ is the number of CPU cycles required to completely process the task $i$, and $F_{r,i}$ and $F_{r,\hat{r}}$ cycles/s are respectively the computation capacity allocated to the task $i$ by $r$ and $\hat{r}$.

From (2), (3) and (4), the total delay to complete the task $i$ is calculated as:

$$
T_{r,i}(t) = T_{r,i}^u(t) + T_{r,\hat{r},i}^m(t) + T_{r,\hat{r},i}^p(t).
$$  

Similar to (5), the cost for processing task $i$ is given by:

$$
c_{r,\hat{r},i}(t) = c_{r,\hat{r},i}^u(t) + c_{r,\hat{r},i}^m(t) + c_{r,\hat{r},i}^p(t), \quad \forall r \in \mathbb{R} \cup \{BS\}, \quad \forall \hat{r} \in \mathbb{R} \cup \{BS\} \setminus \{r\},
$$

where $c_{r,\hat{r},i}^u(t)$, $c_{r,\hat{r},i}^m(t)$ and $c_{r,\hat{r},i}^p(t)$ are respectively the costs of task uploading, task migrating and task processing.

Specifically,

$$
c_{r,\hat{r},i}^p(t) = \delta_{r,i}^p \cdot B_{r,i}(t), \quad \forall r \in \mathbb{R} \cup \{BS\}.
$$

where $\delta_{r,i}^p > 0$ is the communication cost.

After the task $i$ is uploaded to $r$, a service entity hosted at $r$ will handle the task $i$. This entity is migrated from $r$ to $\hat{r} \in \mathbb{R} \cup \{BS\} \setminus \{r\}$ if the task $i$ is not completely processed before the vehicle $i$ leaves the coverage area of $r$. To migrate the service entity from $r$ to $\hat{r}$, the vehicle $i$ incurs the following cost [20]:

$$
e_{r,\hat{r},i}(t) = x_{r,\hat{r},i}(t) \cdot \delta_{r,i}^m \cdot \theta, \quad \forall r \in \mathbb{R} \cup \{BS\}, \quad \forall \hat{r} \in \mathbb{R} \cup \{BS\} \setminus \{r\},
$$

where $\delta_{r,i}^m > 0$ 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:

$$
e_{r,\hat{r},i}(t) = x_{r,i}(t) \cdot \delta_{r,i}^p \cdot F_{r,i} + x_{r,\hat{r},i}(t) \cdot \delta_{\hat{r},i}^p \cdot F_{\hat{r},i}, \quad \forall r \in \mathbb{R} \cup \{BS\}, \quad \forall \hat{r} \in \mathbb{R} \cup \{BS\} \setminus \{r\},
$$

where $\delta_{r,i}^p > 0$ and $\delta_{\hat{r},i}^p > 0$ are the unit computation costs.

III. PROBLEM FORMULATION

There are three constraints that describe the dependence of task assignment on the vehicle mobility, different from [20]. When a task $i \in \mathcal{J}$ is completely executed by an RSU $r \in \mathbb{R} \cup \{BS\}$, 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,

$$
x_{r,i}(t) \left[ T_{r,i}^u(t) - \frac{\hat{d}_{r,i}(t)}{v_i} \right] \leq 0, \quad \forall r \in \mathbb{R} \cup \{BS\}, \quad \forall \hat{r} \in \mathbb{R} \cup \{r\},
$$

where $\hat{d}_{r,i}(t)$ is the distance that the vehicle $i$ travels before leaving the coverage area of $r$.

If the task $i$ is migrated from the RSU $r \in \mathbb{R}$ to another RSU $\hat{r} \in \mathbb{R} \setminus \{r\}$, the delay is instead constrained by:

$$
x_{r,\hat{r},i}(t) \left[ T_{r,\hat{r},i}^m(t) + D_R \cdot h_{r,\hat{r}} \right] \leq 0, \quad \forall r \in \mathbb{R} \cup \{BS\}, \quad \forall \hat{r} \in \mathbb{R} \setminus \{r\}.
$$

If the task $i$ is migrated from the BS to an RSU $\hat{r} \in \mathbb{R}$, the delay is then constrained by:

$$
x_{r,\hat{r},i}(t) \left[ T_{r,\hat{r},i}^m(t) + 2R_R + D_R \cdot h_{r,\hat{r}} \right] \leq 0,
$$
\[ \forall i \in I, r \in \{ \text{BS} \}, \forall \hat{r}, \hat{i} \in \mathcal{R}, \quad (12) \]

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 in vehicular edge computing is thus formulated as the following BNLP problem.

\[
\min_{x_{r,i}(t) \forall r \in \mathcal{R}, i \in I} \sum_{r \in \mathcal{R}} \sum_{i \in I} \left[ \beta \cdot T_{r,i}(t) + \gamma \cdot c_{r,i}(t) \right] \quad (13a)
\]

\[
\text{s.t.} \quad (5), (6), (10), (11), (12) \quad (13b)
\]

\[
\sum_{i \in I} x_{r,i}(t) \cdot F_{r,i} \leq F_r^{\max} \quad \forall r \in \mathcal{R}, \quad (13c)
\]

\[
\sum_{i \in I} \sum_{r \in \mathcal{R}} x_{r,i}(t) \cdot F_{\hat{r},i} \leq F_{\hat{r}}^{\max}
\]

\[
\forall \hat{r} \in \mathcal{R} \cup \{ \text{BS} \} \setminus \{ r \}, \quad (13d)
\]

\[
\sum_{i \in I} x_{r,i}(t) \cdot F_{r,i} + \sum_{i \in I} \sum_{r \in \mathcal{R} \setminus \{ r \}} x_{r,\hat{r},i}(t)
\]

\[
\times F_{\hat{r},i} \leq F_r^{\max} \quad \forall r \in \mathcal{R}, \quad (13e)
\]

\[
x_{r,i}(t) + \sum_{\hat{r} \in \mathcal{R} \setminus \{ \text{BS} \} \setminus \{ r \}} x_{r,\hat{r},i}(t) = 1
\]

\[
\forall r \in \mathcal{R}, \forall i \in I, \quad (13f)
\]

\[
x_{r,i}(t), x_{r,\hat{r},i}(t) \in \{ 0, 1 \} \quad \forall i \in I,
\]

\[
\forall r \in \mathcal{R} \cup \{ \text{BS} \}, \forall \hat{r} \in \mathcal{R} \cup \{ \text{BS} \} \setminus \{ r \}, \quad (13g)
\]

where \( \beta > 0 \) and \( \gamma > 0 \) are the weights to prioritize the delay and the cost, respectively. Constraints (13c), (13d) and (13e) show that the computation capacity assigned to a task is upper-bounded by the maximum computation capacities \( F_r^{\max} \) or \( F_{\hat{r}}^{\max} \). Constraint (13f) enforces that an arbitrary task \( i \) must be processed by one of the RSUs and the BS.

IV. PROPOSED MULTI-AGENT 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 [29], [30]. This procedure is well-known for its low complexity and provable convergence when making decisions in a situation involving multiple stakeholders.

In this paper, we consider that all tasks are homogeneous (see Section I), and the number of tasks is predefined (see Section III). The proposed solution is readily applicable to scenarios where tasks have different deadlines or different sizes, or the number of tasks varies over time. We further introduce a forgetting factor in the learning algorithm to enable fast convergence—an essential requirement in a fast-changing environment due to highly-mobile learning agents (i.e., vehicles). Using simulations with realistic network settings, we will later show that our solution adapts and converges much faster than existing approaches, especially as the number of participating vehicles (i.e., tasks) increases.

A. Game Reformulation for Task Assignment

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 reach an optimal solution. To ensure convergence of the learning solution at the optimum point for all the requesting vehicles, we designate the BS as a central operator. After all the requesting vehicles have played their respective actions, the operator updates each vehicle with the actions chosen by the others. Fig. 2 shows the flowchart of our proposed task assignment scheme.

Specifically, we model the task assignment in vehicular edge computing (13) as a repeated game \( \mathcal{G} = (\mathcal{J}, \mathcal{A}, \mathcal{U}) \), where the agents aim to minimize the long-run average delay and the cost to process the tasks presented by the requesting vehicles. In this model, the (finite) set of requesting vehicles \( \mathcal{J} = \{ 1, 2, \ldots, |\mathcal{J}| \} \) is regarded as the set of agents, and each agent \( i \in \mathcal{J} \) has a set of finite actions \( \mathcal{A}_i \). The vehicles can decide to offload their tasks to one of the RSUs in the set \( \mathcal{R} \) or to the BS. As such, the finite action set \( \mathcal{A}_i \) of an agent \( i \in \mathcal{J} \) consists of the set \( \mathcal{R} \) and the BS, that is, \( \mathcal{A}_i = \mathcal{R} \cup \{ \text{BS} \} \). We denote by \( \mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_{|\mathcal{J}|} \) the set of joint actions of all agents. Let \( \mathcal{U} = \{ u_1, u_2, \ldots, u_{|\mathcal{J}|} \} \) denote the set of utility functions of all the agents.

The objective of our formulated problem (13) is to minimize the delay and cost for processing tasks of the vehicles. Based on the objective function (13a), the utility function of an agent \( i \in \mathcal{J} \) at a time \( t \) resulting from a given action \( a_i^{(t)} = r \in \mathcal{A}_i \) is designed as:

\[
u_i^{(t)}(a_i^{(t)}, a_i^{(t)}) = - \left[ \beta \cdot T_{a_i^{(t)}, a_i^{(t)}}(t) + \gamma \cdot c_{a_i^{(t)}, a_i^{(t)}}(t) \right]. \quad (14)\]
where $a_i^{(t)}$ denotes the vector of RSU/BS actions decided by the $|\mathcal{A}_i| \{j\}$ agents. Here, if action $a_i^{(t)}$ of agent $i$ satisfies all constraints in problem (13), we calculate the parameters $T_{a_i^{(0)}|a_i^{(t)}}(t)$ and $c_{a_i^{(t)}|a_i^{(t)}}(t)$ using (5) and (6), respectively. Otherwise, $T_{a_i^{(0)}|a_i^{(t)}}(t) = \infty$ and $c_{a_i^{(t)}|a_i^{(t)}}(t) = \infty$ where $\infty$ stands for a large positive value pre-assigned. Under this utility model, each agent $i$ obtains an agent-specific payoff depending on the joint action profile $(a_1^{(t)}, a_2^{(t)})$ over all agents. Here, maximizing the sum of all the agents’ utilities would result in minimizing the objective function in problem (13).

### B. Definition of 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 $\mathcal{G}$ is a correlated equilibrium (CE) [31], [32]. A probability distribution $\psi$ defined on $\mathcal{A}$ is said to be a CE if for all agent $i \in \mathcal{I}$, for all $a_i \in \mathcal{A}_i$ and for every pair of action $j, k \in \mathcal{A}_i$, it holds true that

$$\sum_{a_i \in \mathcal{A}_i} \psi(j, a_i) [u_i(k, a_i) - u_i(j, a_i)] \leq 0.$$  \hspace{1cm} (15)

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

### C. RM-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 [30]. The key idea is to adjust the agent’s action probability to be proportional to the “regrets” for not having played other actions. Specifically, for any two actions $j \neq k \in \mathcal{A}_i$ at any time $t$, the cumulative regret of an agent $i$ up to the time $t$ for not playing action $k$ instead of its actually played action $a_i^{(t)}$ is

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

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

The regret can be recursively expressed as:

$$\bar{D}_i^{(t)}(j, k) = \left( 1 - \frac{1}{t} \right) \bar{D}_i^{(t-1)}(j, k) + \frac{1}{t} \bar{D}_i^{(t)}(j, k),$$ \hspace{1cm} (16)

where $\bar{D}_i^{(t)}(j, k) = \mathbb{I}\{a_i^{(t)} = j\} [u_i^{(t)}(k, a_i^{(t)}) - u_i^{(t)}(j, a_i^{(t)})]$ denotes the instantaneous regret by the agent $i$ 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 by adding a correction term based on the new instantaneous regret. As a stochastic approximation method, (16), although resulting in almost surely convergence, can be quite slow. This is especially true in a dynamic environment, where agent’s utility changes with time. This 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 $\bar{D}_i^{(t)}(j, k)$ as:

$$\bar{D}_i^{(t)}(j, k) = \lambda \bar{D}_i^{(t-1)}(j, k) + (1 - \lambda) \bar{D}_i^{(t)}(j, k),$$ \hspace{1cm} (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 agent then independently chooses its next action according to the following probabilities:

$$\pi_i^{(t+1)}(j) = 1 - \sum_{k \neq j} \pi_i^{(t+1)}(k).$$

### Algorithm 1: Multi-Agent RM Learning-Based Task Assignment Algorithm.

1. **Input:** Actions selected by agents at time $t$.
2. **Output:** Probability that agents will play their actions in the next time $t + 1$.
3. **Initialization:** Each agent $i$ initializes its action selection policy with a uniform strategy $\pi_i^{(1)}(j) \leftarrow \frac{1}{|\mathcal{A}_i|} \forall j \in \mathcal{A}_i$
4. **for** $t = 1, 2, \ldots$ **do**
   5.  Each agent $i \in \mathcal{I}$ independently runs the following procedure to decide its action over time.
   6.  **Step 1 (Action selection):**
       1. Agent $i$ samples an action $a_i^{(t)} = j \in \mathcal{A}_i$ from its probability distribution of action selection $\pi_i^{(t)}$.
   7. The BS updates the chosen action of agent $i$ to all other agents.
   8. **Step 2 (RM learning):**
       1. Agent $i$ receives a utility as a result of its chosen action $u_i^{(t)}(j, a_i^{(t)})$ computed by (14).
   10. **for** $k \in \mathcal{A}_i \setminus \{j\}$ **do**
       11. Agent $i$ would play action $j$ in the next time with the remaining probability $\pi_i^{(t+1)}(j) = 1 - \sum_{k \neq j} \pi_i^{(t+1)}(k)$.

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

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
If action combination $(I, A)$ agent chooses their actions according to Algorithm 1, then the joint empirical distribution of action profiles converges almost surely to the Correlated Equilibrium set of the game $\mathcal{G}$ as $t \to \infty$.\footnote{The proof is given in the appendix.}

Remark: Our proposed algorithm follows the RM based online learning rule, which has shown to be able to compute an CE almost surely in the limit, and an approximate $\epsilon$-CE with rate $(1/\sqrt{\psi})$ with high probability [29], where $\psi$ is the number of iterations at which the algorithm is run. Furthermore, the complexity of finding a CE using RM is known to be only polynomial in the number of agents, the upper bound on the number of actions per agent, and the approximation error $\epsilon^{-1}$ (rather than exponential as in finding a Nash equilibrium) [33], [34]. It will be interesting to consider the question of how well our proposed algorithm scales with regards to the size of the problem and the number of available actions for each agent to choose from. We will discuss the computational scalability of our approach in details in the following subsection.

D. Computational Complexity of the Proposed Algorithm

We discuss here the computational complexity aspects of the proposed Algorithm 1. A multi-agent normal-form game can be represented with an utility set $\mathcal{U}$ in which $u_i(a_i, a_j)$ specifies an individual utility for agent $i$ under a combination of actions $(a_i, a_j)$. The number of action combination grows exponentially in the number of agents, i.e., a game with $|\mathcal{I}|$ agents and at most $|A_i|$ actions per agent $i$ ($i \in \mathcal{I}$) has $|A_i|^{|\mathcal{I}|}$ action combination in total. Each agent has one individual utility for each action combination and thus it requires $|\mathcal{I}| \times |A_i|^{|\mathcal{I}|}$ integer numbers to represent all agents’ utilities. Therefore, the complexity of searching over the space of all possible action combination is $\mathcal{O}(|\mathcal{I}| \times |A_i|^{|\mathcal{I}|})$, which is exponential in the size of the input and thus is generally intractable both for storing a game and computing its solution.

One important aspect of our proposed distributed mechanism is that the computation can be performed in parallel by all the agent in each decision stage of the game. This feature reduces the computation time of the algorithm dramatically and thus can offer many advantages over traditional centralised solutions, such as enhanced speed, scalability, and efficiency. In particular, our algorithm allows every agent independently decides its action in a distributed fashion based on its own utility observation. Using our method, at each time step $t$, every agent $i$ announces its chosen action to a BS which is subsequently broadcast to all agents.

Fig. 3. Deployment of 1 BS and 2 RSUs along a six-lane highway in Scenario 1.

Fig. 4. Deployment of 1 BS and 10 RSUs along a six-lane highway in Scenario 2.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE II
PERFORMANCE COMPARISON OF THE FOUR ALGORITHMS IN THE TWO SCENARIOS CONSIDERED. HERE, WE USE ONLY ONE CPU CORE TO IMPLEMENT THE FOUR ALGORITHMS

| Scenario 1 | Scenario 2 |
|------------|------------|
| Forgetting factor ($\lambda$) | N/A | N/A | 0.5 |
| Minimum sum of delay and cost | $\approx 1.64 \times 10^3$ | $\approx 1.64 \times 10^3$ | $\approx 5.71 \times 10^3$ | $\approx 1.64 \times 10^3$ |
| Computation time (s) | $\approx 2.47 \times 10^4$ | 0.26 | 4.88 | 0.1 |

respectively. Here, the vehicles travel along the highway at constant speeds while they never overtake each other.

For a comprehensive comparison, we have to benchmark Algorithm 1 against the most relevant related works in the literature. Because [19] and [20] do not design task assignment algorithms for allocating multiple autonomous vehicles’ tasks to servers with respect to vehicle movement in highway scenarios, we have to compare Algorithm 1 with the following three algorithms.

1) **Exhaustive Search (ES)**: A centralized algorithm where a central operator collects all network information and finds the globally optimal solution using exhaustive search.

2) **Traditional Regret-Matching (TRM) scheme [22]**: A distributed algorithm where each agent selects an action on the basis of its regret value, and the regret values are not calculated with respect to changes in the vehicular network.

3) **Reinforcement Learning with Network-Assisted Feedback (RLNF) scheme [21]**: A distributed algorithm where each agent selects its action without knowing global network conditions.

It is noted that the TRM and RLNF schemes are also based on RM learning; nevertheless, they have not employed a forgetting factor to update regret values, unlike Algorithm 1.

To demonstrate that Algorithm 1 works effectively in not only small-scale but also large-scale environments, we evaluate all four algorithms in two scenarios. Specifically, in Scenario 1, there are 3 servers deployed along the highway while 10 vehicles request to complete their tasks. In contrast to Scenario 1, we increase the number of servers and requesting vehicles to 11 and 100, respectively, in Scenario 2. On the other hand, in both scenarios, the requesting vehicles in lanes $\{1, 4\}$, $\{2, 5\}$ and $\{3, 6\}$ are moving at a speed of 90, 100 and 120 km/h, respectively. In addition, the inter-RSU distance is set as 3 km.

B. Simulation Results

Table II compares the performance of Algorithm 1 with the three benchmark schemes. Here, the error tolerance of the four algorithms is set as 0.1. According to the objective function in (13), Algorithm 1 aims to minimize the total delay and cost for task processing; thus, we select the sum of delay and cost as a performance metric. Here, the total delay plus cost and the computation time for task completion are shown for two scenarios, as shown in Figs. 5 and 6.

In Scenario 1, by using 3 servers, Algorithm 1, ES and TRM complete 10 tasks with the lowest total delay plus cost of $1.64 \times 10^3$. In contrast to the three algorithms, the agents in RLNF are not able to select actions to achieve an equilibrium solution because they do not know each other’s strategy. Importantly, Algorithm 1 finds the best solution with the smallest computation time of 0.1 s (12 iterations) compared to ES and TRM. Hereby, Algorithm 1 can be applicable to the real deployment scenarios as it satisfies the latency requirement in vehicular applications, i.e. from 0.1 s to 0.5 s [36], [37].

[Online]. Available: https://github.com/Long-UoA/Multi-Agent-RM-based-Task-Assignment-in-VEC.git
In Scenario 2, the number of servers and tasks is increased up to 11 and 100, respectively. Given the specifications of a typical PC, it is impossible for ES to iterate through \((11^{100} \approx 1.38 \times 10^{104})\) potential solutions. Of the remaining three algorithms, Algorithm 1 converges to the CE solution within the shortest computation time of 45.87 s (907 iterations), giving the lowest total delay plus cost of \(1.7 \times 10^4\). In general, a CE solution might not necessarily be the optimal solution for the BNLP problem (13). Here, we show through experimental results that in most realistic networks, the gap between CE and optimal solution is small (almost negligible) — illustrating that Algorithm 1 provides a good trade off between convergence speed and optimality.

The fast convergence behaviour of the proposed scheme is essential for a dynamic environment in which ITS applications operate. Figs. 7, 8 and 9 illustrate how the convergence of Algorithm 1 depends on the forgetting factor \(\lambda\). In addition, Figs. 8 and 9 show the impact of \(\lambda\) on delay and cost, respectively. As seen, the fastest convergence occurs when \(\lambda\) decreases from 0.9999 to 0.5. According to (17), the cumulative regret of each agent in Algorithm 1 is updated with respect to both their outdated regret and instantaneous regret. When the value of the forgetting factor is high, i.e., 0.9999 or 0.99, Algorithm 1 mainly relies on the outdated regret to compute the cumulative regret. As such, Algorithm 1 adapts slowly to changes in the operating vehicular environment. On the other hand, at \(\lambda = 0.5\), the cumulative regret of an agent is updated with respect to their instantaneous regret rather than their outdated regret. Hence, Algorithm 1 converges to the CE solution faster. Furthermore, Fig. 10 shows that such fast convergence is always maintained irrespective of the number of participating agents. Here, we increase the number of agents from 25 to 100 without any changes in the number of servers or the inter-server distance. While such an increase results in a growth in the total delay and cost, it does not affect the convergence speed of Algorithm 1 which depends only on the forgetting factor \(\lambda\).

To quantify fairness in terms of utility among agents (autonomous vehicles), we make use of Jain’s fairness index proposed in [38] as follows:

\[
J_F = \frac{\sum_{i \in [3]} u_i(t) \left( a_i(t), \tilde{a}_i(t) \right)^2}{\left| \sum_{i \in [3]} u_i(t) \left( a_i(t), \tilde{a}_i(t) \right) \right|^2},
\]

where \(\tilde{a}_i(t) = 1\) for all participating agents. In addition, the utility fairness among the agents would be maintained when the approximate value of \(J_F\) is 1. As shown in Fig. 11, Algorithm 1 achieves the fairness \(J_F\) close to 1, e.g. 0.79, even though the autonomous vehicle density is high. Here, to calculate \(J_F\), we employ the value of agents’ utility.
when Algorithm 1 converges at the correlated equilibrium. In Algorithm 1, given the selected actions of other agents, each agent attempts to play an action not to regret that decision. The CE solution is achieved only when all agents no longer want to deviate from their strategies. In such an event, the utilities obtained by the agents can no longer be improved, which means the utility fairness among the agents is maintained by Algorithm 1.

Fig. 12 demonstrates that Algorithm 1 adapts quickly to the environment changes caused by the agents’ mobility. Here, when the agents move at high speeds, it causes a decrease in the duration when they pass through an RSU, or they will enter the next RSUs’ coverage range. Thus, the number of agents’ actions that are able to both minimize the sum of delay and cost, and satisfy constraints in (13) is reduced significantly. Certainly, it would be much less than that of the scenario in which the agents traverse the highway with lower speeds. As a result, the convergence speed of Algorithm 1 in the former is quicker than that in the latter. In particular, Fig. 13 shows the influence of RSU deployment on the performance of our proposed scheme. With the shortest distance between two consecutive RSUs (i.e., \( D_R = 3 \) km), the RSUs are distributed densely along the six-lane highway. It leads to the lowest total delay and cost achieved by Algorithm 1. Extending the inter-RSU distance will cause an increase in the total delay and cost. That is because the processing delay of a task depends on the inter-RSU distance according to constraints (11) and (12). Given the larger inter-RSU distance of 9 km, the task migration time (and thus the processing time) is longer.

VI. CONCLUSION

To tackle the challenge of constrained computational resources within Vehicular Edge Computing (VEC), we propose an innovative task assignment scheme. This scheme orchestrates the seamless migration of vehicles’ tasks across a network of multiple servers, intelligently aligned with their distinct mobility patterns.

To this aim, we present a novel approach through the formulation of a Bayesian Non-Linear Programming (BNLP) problem. The problem formulation minimizes both total delay and cost implications stemming from the participation of various vehicles. Furthermore, we present a multi-agent Reinforcement Learning (RL) based algorithm. The theoretical underpinnings of this algorithm ensure its convergence towards the Correlated Equilibrium (CE) solution.

Validation and performance evaluation are accomplished through extensive simulations, which underscore the advantages offered by our proposed algorithm over existing approaches. The simulation code is publicly available\(^4\).

APPENDIX A

PROOF OF THEOREM 1

Proof: For notational convenience, let us drop the subscript \( i \) and define the following Lyapunov function:\(^5\)

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

\( \cdot^+ \) denotes the positive part of the argument. Thus, the update rule for the algorithm is given by

\[
\vec{D}(n+1) = \text{argmin} [P(\vec{D})]
\]

which is equivalent to the solution of the Bayes' rule.

\(^4\)[Online]. Available: https://github.com/Long-UoA/Multi-Agent-RM-based-Task-Assignment-in-VEC.git.

\(^5\)dist\((x, A) = \min \{\|x - a\| : a \in A\}, \) where \(\| \cdot \|\) is the Euclidean norm.
where $\mathbb{R}^-$ represents the negative orthant. Taking the time-derivative of (20) yields
\[
\frac{d}{dt} P(\bar{D}) = \sum_{j,k} |\bar{D}(j,k)|^+ \times \frac{d}{dt} \bar{D}(j,k).
\] (21)

First, we find $d\bar{D}(j,k)/dt$ by rewriting (17) as:
\[
\bar{D}^{(t)}(j,k) = \bar{D}^{(t-1)}(j,k) + (1 - \lambda) \left\{ D^{(t)}(j,k) - \bar{D}^{(t-1)}(j,k) \right\}
\]
\[
= \bar{D}^{(t-1)}(j,k) + (1 - \lambda) \left\{ u(k, \cdot) - u(j, \cdot) \right\} \mathbb{1}\{a_i^{(t)} = j\}
\]
\[- \bar{D}^{(t-1)}(j,k) \}
\] (22)

Let $\epsilon = 1 - \lambda$ be a constant step size. It can be seen that (22) has the form of a constant step size stochastic approximation algorithm $\theta_{k+1} = \theta_k + \epsilon H(\theta_k, x_k)$ and satisfies [39, Th. 17.1.1]. Thus, its dynamics can be characterized by an ordinary differential equation (see [39, Ch. 17] for further details). This means the system can be approximated by replacing $x_k$ with its expected value. By applying [39, Theorem 17.1.1], $\bar{R}_k(j,k)$ converges weakly (in distribution) to the averaged system corresponding to (22). As such,
\[
\frac{d}{dt} \bar{D}(j,k) = \mathbb{E} \left\{ [u(k, \cdot) - u(j, \cdot)] \mathbb{1}\{a_i^{(t)} = j\} - \bar{D}(j,k) \right\}
\]
\[= [u(k, \cdot) - u(j, \cdot)] \pi(j) - \bar{D}(j,k).
\] (23)

Next, replacing $d\bar{D}(j,k)/dt$ from (23) into (21) gives:
\[
\frac{d}{dt} P(\bar{D}) = \sum_{j,k} |\bar{D}(j,k)|^+ [u(k, t) - u(j, t)] \pi(j)
\]
\[- \sum_{j,k} |\bar{D}(j,k)|^+ \times \bar{D}(j,k)
\]
\[
\leq 2G\delta |A_i| \sum_{j,k} |\bar{D}(j,k)|^+ - 2P(\bar{D}).
\] (24)

where $G$ is an upper bound on $|u(\cdot)|$, $0 \leq \delta \leq 1$, and $|A_i|$ is the cardinality of the set $A_i$ (the set of actions of a player $i$). Note that in the last equality of (24), we have used the following two lemmas:

1. \[\sum_{j,k} |\bar{D}(j,k)|^+ \bar{D}(j,k) = 2P(\bar{D})\] (immediate from Eq. (20))

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

Finally, it follows from (24) that by assuming there exists some positive constant $\kappa > 0$ such that $|\bar{D}(j,k)|^+ \geq \kappa > 0$, one can choose a sufficiently small $\delta > 0$ such that (see [41] for more details)
\[
\frac{d}{dt} P(\bar{D}) \leq -P(\bar{D}).
\]

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

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:
\[
\phi_t \left( j, a_i^{(t)} \right) = \phi_{t-1} \left( j, a_i^{(t-1)} \right)
\]
\[+ \epsilon \left[ \mathbb{1}\{a_i^{(t)} = j\} - \phi_{t-1} \left( j, a_i^{(t-1)} \right) \right]
\]
\[= \epsilon \sum_{\tau \leq t} (1 - \epsilon)^{t-\tau} \mathbb{1}\{a_i^{(t)} = j\}.
\] (25)

The elements of the regret matrix in (16) can be rewritten as follows
\[
\bar{D}_i^{(t)}(j,k) = \epsilon \sum_{\tau \leq t} (1 - \epsilon)^{t-\tau} [u_i(k, \cdot) - u_i(j, \cdot)] \mathbb{1}\{a_i^{(t)} = j\}
\]
\[= \epsilon \sum_{\tau \leq t} (1 - \epsilon)^{t-\tau} \left[ \mathbb{1}\{a_i^{(t)} = j\} \right] [u_i(k, \cdot) - u_i(j, \cdot)]
\]
\[= \epsilon \sum_{\tau \leq t} (1 - \epsilon)^{t-\tau} \left[ \mathbb{1}\{a_i^{(t)} = j\} \right] [u_i(k, \cdot) - u_i(j, \cdot)].
\] (26)

On the last line of (26), we have substituted $\phi_t(j, a_i^{(t)})$ from (25). Finally, on any convergent subsequence $\lim_{t \to \infty} \phi_t \to \psi$, we have:
\[
\lim_{t \to \infty} \bar{D}_i^{(t)}(j,k) = \sum_{a_i} \psi(j, a_i) [u_i(k, a_i) - u_i(j, a_i)] \leq 0.
\] (27)

Comparing (27) with the definition of Correlated Equilibrium in (15) completes the proof. \hfill \square

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. 10190–10203, Nov. 2018.

[2] L. Silva et al., “Computing paradigms in emerging vehicular environments: A review,” IEEE/CAA J. Automatica Sinica, vol. 8, no. 3, pp. 491–511, Mar. 2021.

[3] Z. Ning et al., “Joint computing and caching in 5G-Envisioned internet of vehicles: A deep reinforcement learning-based traffic control system,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 8, pp. 5201–5212, Aug. 2021.

[4] A. Girma et al., “IoT-Enabled autonomous system collaboration for disaster-area management,” IEEE/CAA J. Automatica Sinica, vol. 7, no. 5, pp. 1249–1262, Sep. 2020.

[5] H. Chang, Y. Chen, B. Zhang, and D. Doermann, “Multi-UAV mobile edge computing and path planning platform based on reinforcement learning,” IEEE Trans. Emerg. Topics Comput. Intell., vol. 6, no. 3, pp. 489–498, Jun. 2022.

[6] M. Yi, P. Yang, M. Chen, and N. T. Loc, “A DRL-Driven intelligent joint optimization strategy for computation offloading and resource allocation in ubiquitous edge IoT systems,” IEEE Trans. Emerg. Topics Comput. Intell., vol. 7, no. 1, pp. 39–54, Feb. 2023.

[7] Transport for NSW, “Traffic statistics.” Accessed: Dec. 1, 2022. [Online]. Available: https://www.rms.nsw.gov.au/about/corporaterepublications/statistics/traffic-statistics/index.html

[8] J. Bi, H. Yuan, S. Duamua, M. Zhou, and A. Abusorrah, “Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization,” IEEE Internet Things J., vol. 8, no. 5, pp. 3774–3785, Mar. 2021.

[9] H. Yuan and M. Zhou, “Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
systems,” IEEE Trans. Automat. Sci. Eng., vol. 18, no. 3, pp. 1277–1287, Jul. 2021.
[10] F. Zeng, Q. Chen, L. Meng, and J. Wu, “Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 6, pp. 3247–3257, Jun. 2021.
[11] X. Bai, W. Yan, and S. S. Ge, “Efficient task assignment for multiple vehicles with partially unreachable target locations,” IEEE Internet Things J., vol. 8, no. 5, pp. 3730–3742, Mar. 2021.
[12] Z. Zhang, M. Zhou, and J. Wang, “Construction-based optimization approaches to airborne crew rostering problem,” IEEE Trans. Automat. Sci. Eng., vol. 17, no. 3, pp. 1399–1409, Jul. 2020.
[13] Z. Zong, M. Zheng, Y. Li, and D. Jin, “MAPDP: Cooperative multi-agent reinforcement learning to solve pickup and delivery problems,” in Proc. AAAI Conf. Artif. Intell., 2022, pp. 9980–9988.
[14] S. Sarkar et al., “Multi-agent reinforcement learning controller to maximize energy efficiency for multi-generator industrial wave energy converter,” in Proc. AAAI Conf. Artif. Intell., 2022, pp. 12135–12144.
[15] W. Zhan et al., “Deep-reinforcement-Learning-Based offloading scheduling for vehicular edge computing,” IEEE Internet Things J., vol. 7, no. 6, pp. 5449–5465, Jun. 2020.
[16] L. Qian, Y. Wu, N. Yu, F. Jiang, H. Zhou, and T. Q. Quek, “Learning driven NOMA assisted vehicular edge computing via underlay spectrum sharing,” IEEE Trans. Veh. Technol., vol. 70, no. 1, pp. 977–992, Jan. 2021.
[17] Y. Ju et al., “Joint secure offloading and resource allocation for vehicular edge computing network: A multi-agent deep reinforcement learning approach,” IEEE Trans. Intell. Transp. Syst., vol. 24, no. 5, pp. 5555–5569, May 2023.
[18] H. Peng and X. S. Shen, “Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks,” IEEE Netw., vol. 34, no. 4, pp. 916–923, Jul.–Aug. 2020.
[19] 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. 171140–171153, 2020.
[20] 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, Aug. 2020.
[21] 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, Sep. 2017.
[22] 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, Jan. 2020.
[23] X. Xu and Q. Zhao, “Distributed no-regret learning in multiagent systems: Challenges and recent developments,” IEEE Signal Process. Mag., vol. 37, no. 3, pp. 84–91, May 2020.
[24] F. Sentenac, E. Bourier, and V. Perchet, “Decentralized learning in online queueing systems,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2021, pp. 18501–18512.
[25] G. E. Flaspohler et al., “Online learning with optimism and delay,” in Proc. 38th Int. Conf. Mach. Learn., 2021, pp. 3363–3373.
[26] I. Bistritz and N. Bambos, “Cooperative multi-player bandit optimization,” in Proc. Int. Conf. Neural Inf. Process. Syst., H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020, pp. 2016–2027.
[27] C. Daskalakis and N. Golowich, “Fast rates for nonparametric online learning: From realizability to learning in games,” in Proc. 54th Annu. ACM SIGACT Symp. Theory Comput., 2022, pp. 846–859.
[28] I. Anagnostides, C. Daskalakis, G. Farina, M. Fishelson, N. Golowich, and T. Sandholm, “Near-optimal no-regret learning for correlated equilibria in multi-player general-sum games,” in Proc. 54th Annu. ACM SIGACT Symp. Theory Comput., 2022, pp. 734–749.
[29] S. Hart and A. Mas-Colell, “A simple adaptive procedure leading to correlated equilibrium,” Econometrica, vol. 68, no. 5, pp. 1127–1150, 2000.

Bach Long Nguyen received the Ph.D. degree in engineering from The University of Newcastle, Callaghan, NSW, Australia, in 2021. Since then, he has been a Visiting Scholar with Monash University, Clayton, VIC, Australia, and a Research Fellow with The University of Adelaide, Adelaide, SA, Australia. His top three Ph.D. theses published in Journal and Proceedings of Royal Society of New South Wales, Australia. His research interests include optimisation and learning-based algorithms for autonomous multi-agent systems, maritime transport and intelligent transportation systems.

Duong D. Nguyen received the B.Sc. (Hons.) degree in electronic communication systems from the University of Plymouth, Plymouth, U.K., in 2008, the M.Sc. degree in mobile and personal communication from King’s College London, London, U.K., in 2009, and the Ph.D. degree in engineering from The University of Adelaide, Adelaide, SA, Australia, in 2018. Since then, he has been a Postdoctoral Research Fellow with The University of Adelaide. His research interests include game theory models and decision-making algorithms for autonomous systems.

Hung X. Nguyen received the Ph.D. degree in computer and communication sciences from Swiss Federal Institute of Technology Lausanne, Lausanne, Switzerland. He is currently an Associate Professor in computer science with the University of Adelaide, Australia, where he has been since 2009. His current research interests include models and algorithms for improving network performance, security, and resiliency, especially for IoT and wireless networks.
Duy T. Ngo (Senior Member, IEEE) received the B.Eng. (with first-class Honors, and the University Medal) degree in telecommunication engineering from The University of New South Wales, Sydney, NSW, Australia, in 2007, the M.Sc. degree in electrical engineering (communication) from the University of Alberta, Edmonton, AB, Canada, in 2009, and the Ph.D. degree in electrical engineering from McGill University, Montreal, QC, Canada, in 2013. He is currently an Associate Professor with the School of Engineering, The University of Newcastle, Callaghan, NSW, Australia. His research interests include wireless communications, intelligent transportation systems, and machine learning applications.

Markus Wagner received the Ph.D. degree from the Max Planck Institute for Informatics, Saarbruecken, Germany, and with the University of Adelaide, Adelaide, SA, Australia. He is currently an Associate Professor with Monash University, Melbourne, VIC, Australia. For the outcomes of his studies, he was the recipient of the university’s Doctoral Research Medal - the first for his school - and four best paper awards as well as a Humies Gold Award. He has been the Program Committee Member more than 80 times, and he has written more than 150 articles with more than 200 different co-authors. His research interests include mathematical runtime analysis of heuristic optimisation algorithms and theory-guided algorithm design to applications of heuristic methods to renewable energy production, professional team cycling and software engineering. He is an ACM Lifetime Member, is on SIEVO’s Executive Board and is the first ever Sustainability Officer. He has contributed to GECCOs as Workshop Chair and Competition Chair, and he has chaired several education-related committees within the IEEE CIS, where he also was the Founding Chair of two task forces.