Hybrid Federated and Multi-agent DRL-Based Resource Allocation in Digital Twin-IoV Networks

Bishmita Hazarika, Anal Paul, and Keshav Singh(✉)

Institute of Communications Engineering, National Sun Yat-sen University, Kaohsiung 804, Kaohsiung, Taiwan
apaul@ieee.org, keshav.singh@mail.nsysu.edu.tw

Abstract. This study introduces a combined machine learning strategy for optimizing resource distribution within a digital twin (DT) setup, aimed at offloading tasks in UAV-supported Internet-of-Vehicles networks. By merging asynchronous federated learning with multi-agent deep reinfo- rcement learning, we aim to boost the rate of task completion while simultaneously reducing consumed energy and delay, and consequently improving the system's overall performance. We introduce a DT architecture within an IoV network supported by UAV for V2V and V2I task transitions accommodating three modes of processing tasks and a trio of task categories. We then formulate a problem focusing on elevating the overall effectiveness of the system, aiming to reduce both delay and energy usage. Addressing this complex challenge, our solution introduces a multi-agent DRL algorithm designated as MARS, dedicated to the effective distribution of resources within the IoV network supported by digital twins. This algorithm is refined through a blended AFL method, which we refer to as HAFL. MARS enhances the distribution of resources across various computational settings to augment the total system efficacy. Finally, comprehensive simulations affirm our approach's superiority, bench- marked against a variety of established methodologies.

Keywords: Asynchronous federated learning · Deep reinforcement learning · Internet of Vehicles

1 Introduction

The Internet of Vehicles (IoV) signifies a leap forward in improving the efficiency and safety of transportation systems. Nonetheless, the deployment of IoV networks faces obstacles due to their constrained processing power and storage capabilities, which complicates the delivery of superior services. To address this,

This work was supported by the National Science and Technology Council of Taiwan under Grants NSTC 112–2221-E-110–038-MY3.

N.-S. Vo et al. (Eds.): INISCOM 2024, LNICST 595, pp. 109–121, 2024.
https://doi.org/10.1007/978-3-031-67357-3_8

mobile edge computing (MEC) has been proposed as a solution. However, its effectiveness is limited by stationary locations and high installation costs [4]. A promising resolution comes in the form of unmanned aerial vehicles (UAVs), which mitigate MEC's limitations through their agile deployment and support in data transmission. Specifically, UAVs strengthen vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) commun- ications, aiding in latency reduction and MEC server load alleviation. Nonetheless, integrating UAVs with V2V and V2I is not without its challenges, such as potential transmission delays due to extended line-of-sight (LoS) communi- cation links. Despite these hurdles, UAV-supported MEC networks hold immense poten- tial to amplify IoV network efficacy, particularly in fluctuating environments.

Digital twin (DT) technology presents a viable approach to overcoming the difficulties faced by IoV networks, by creating virtual representations of physical objects and facilitating instantaneous tracking for improved decision processes [6,7]. Furthermore, integrating deep reinforcement learning (DRL) with DT-oriented edge networks (DITEN) boosts the precision of forecasts for these systems. The amalgamation of DRL and asynchronous federated learning (AFL) additionally advances the efficiency of resource distribution within DITEN [2]. AFL can overcome the challenge of limited resources by facilitating collaborative learning among distributed devices while reducing transmission time, and energy consump- tion, and maintaining data privacy [3,8]. However, traditional DRL models may be less effective in the complex IoV network. In such cases, multi-agent deep reinforcement learning (MADRL) can perform efficiently by capturing the agent interactions and optimizing their behaviors, thereby, improving network perform- ance [5]. Consequently, the authors in [9] have shown that combining AFL with MADRL can further optimize resource allocation decisions and enhance the efficiency of DT-assisted IoV networks.

Based on the above discussions, we introduce a hybrid approach that merges AFL with multi-agent DRL for effective resource management in DT-enabled IoV ecosystems, particularly for task offloading within UAV-supported IoV netwo- rks. This approach aims to concurrently enhance task completion speeds, lower energy consumption, and decrease delays, thus elevating the system's overall performance. To our knowledge, this study is the first to blend AFL with multi-agent DRL within DT-enhanced IoV settings. The primary contributions of our proposed methodology are outlined as:

- We introduce a UAV-aided IoV network with DT-enabled V2V and V2I offloading, ensuring broad coverage and real-time data updates. Our resource allocation system within this framework categorizes task vehicles, service vehicles, and RSUs, integrating three execution modes and task types.
- Following this, we formulate a problem to enhance the overall performance of the system, specifically by improving task completion efficiency and minimizing both delay and energy consumption. To address this challenge, we introduce MARS, a multi-agent DRL algorithm which stands for **M**ulti-**A**gent **R**esource **S**haring for DT-assisted IoV Networks. MARS utilizes a hybrid AFL approach, HAFL, for its training process.

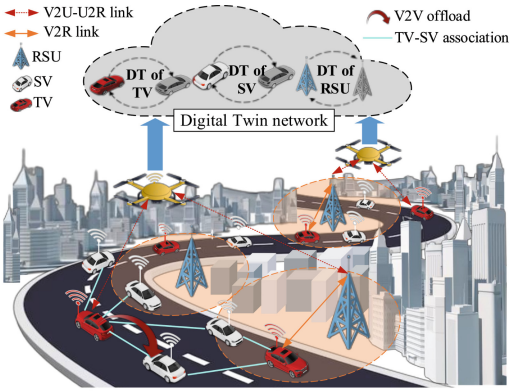

Fig. 1. An illustration of our DT-driven and UAV-supported IoV network.

– Extensive simulations validate our approach against synchronous federated learning (SFL) and single and multi-agent deep deterministic policy gradient (DDPG) algorithms as benchmarks, demonstrating its efficacy.

2 System Model

2.1 System Architecture

This study explores a UAV-supported DITEN-enabled IoV network as shown in Fig. 1) comprising V vehicles, R RSUs, and U UAVs. The network facilitates three types of computation methods: local execution, V2V offloading, and V2I offloading. Within this setup, vehicles are categorized into task vehicles (TVs) and service vehicles (SVs). Here, T TVs offload tasks, whereas S SVs are assigned to handle the processing of these tasks. RSU-based MEC servers cater to diverse delay require- ments. Time management is executed through mobility time slots segmented into multiple periods with N slots each, assuming constant vehicular speeds within slots but allowing for variations across periods. UAVs provide extensive network coverage, facilitating task offloading to RSUs or acting as relays when appropriate RSUs are unavailable. UAVs hover at a fixed altitude ($H > 0$), positioned at $[x_u, y_u, H]$. Distances like $\mathrm{TV_t}$-$\mathrm{SV_s}$, $\mathrm{TV_t}$-$\mathrm{RSU_r}$, $\mathrm{TV_t}$-$\mathrm{UAV_u}$, and $\mathrm{UAV_u}$-$\mathrm{RSU_r}$ are calculated using the Haversine formula. These distan- ces are individually denoted as $\lambda_{t,s}$ for TV to SV, $\lambda_{t,r}$ for TV to RSU, $\lambda_{t,u}$ for TV to UAV, and $\lambda_{r,u}$ for UAV to RSU, respectively. Our focus is on autonomous, non-divisible, delay-sensitive vehicular tasks, classified into three categories based on delay tolerance-

Crucial tasks: Executed locally due to strict delay constraints. **Smaller high-priority tasks**: Delay-sensitive; executed locally if resources permit or offloaded to SVs. **Larger high-priority tasks:** Delay-sensitive; offloaded to RSUs for processing.

2.2 Dynamic DITEN Model

DT layer provides virtual replicas of physical entities like vehicles and RSUs, encompassing DTs for TVs, SVs, and RSUs within the UAV-assisted network. This method addresses possible discrepancies in digital representations by employing vectors $\widetilde{f}^{RSU}[n]$ and $\widetilde{f}^{SV}[n]$ for RSUs and SVs, respectively. These vectors play a vital role in detecting fluctuations in the expected performance of RSUs and SVs for every distinct time slot n. As a result, the digital twins for RSUs, SVs, and TVs at time interval n are denoted as

$$\mathcal{D}^{RSU}[n] = \left\{ \mathcal{F}^{RSU}[n] \pm \widetilde{f}^{RSU}[n], \mathcal{G}^{RSU}[n], \mathcal{L}^{RSU}[n] \right\}, \tag{1}$$

$$\mathcal{D}^{SV}[n] = \left\{ \mathcal{F}^{SV}[n] \pm \widetilde{f}^{SV}[n], \mathcal{G}^{SV}[n], \mathcal{L}^{SV}[n] \right\}, \tag{2}$$

$$\mathcal{D}^{TV}[n] = \left\{ \mathcal{F}^{loc}[n] \pm \widetilde{f}^{loc}, \mathcal{C}[n], \mathcal{D}[n], \mathcal{G}^{TV}[n], \mathcal{L}^{TV}[n], \mathbb{D}[n], z[n], \mathcal{M}[n] \right\}, \tag{3}$$

where $\mathcal{F}^{RSU}[n]$, $\mathcal{F}^{SV}[n]$, and $\mathcal{F}^{loc}[n]$ indicate the estimated computational capac- ities, $\mathcal{G}^{RSU}[n]$ and $\mathcal{G}^{SV}[n]$ provide location information, and $\mathcal{L}^{RSU}[n]$ and $\mathcal{L}^{SV}[n]$ indicate anticipated delay metrics for RSUs and SVs, respectively. The vectors $\mathcal{M}[n]$ and $\mathcal{G}^{TV}[n]$ represent TVs' local models and network topology data, while $\widetilde{f}^{loc}$ shows the potential variance in local processing power in the DT. Additionally, $\mathcal{C}[n]$, $D[n]$, and $\mathcal{L}^{TV}[n]$ correspond to the CPU cycles, task sizes, and tasks' maximum permissible delay, respectively. Hence, a task from TV_t at time-slot n is represented as $c_t[n], d_t[n], \gamma_t[n]$.

2.3 Computation Model

We consider the delay to encompass task execution times, including transmission, queuing, and execution times, while energy consumption is attributed to each task execution.

Local Execution: Local task execution time for ($\mathrm{TV_t}$) is computed as $\overline{\Gamma}_t^{loc}[n] = (c_t[n]d_t[n])/(f_t^{loc}[n])$, with deviations accounted for by $\Delta\Gamma_t^{loc}[n] = (c_t[n]d_t[n]\widetilde{f}_t^{loc}[n])/(f_t^{loc}[n])$. The actual time is $\Gamma_t^{loc}[n] = \overline{\Gamma}_t^{loc}[n] - \Delta\Gamma_t^{loc}[n]$ and energy consumption is $\mathcal{E}_t^{loc}[n] = \kappa_t(f_t^{loc}[n] - \widetilde{f}_t^{loc}[n])^2 d_t[n]c_t[n]$.

V2V Offload: In V2V offloading scenarios, it's assumed that wireless channels are stable. High-priority tasks that cannot be processed locally are transferred to SVs. The rate at which data is sent from $\mathrm{TV_t}$ to $\mathrm{SV_s}$ is calculated as $\mathcal{R}t, s = \mathcal{B}\log_2(1 + \frac{\omega t[n]h_{t,s}[n]}{\sigma^2})$, such that ω_t is the power of transmission of the vehicles, and $h_{t,s}$ denotes the channel from $\mathrm{TV_t}$ to $\mathrm{SV_s}$. This allows for the estimation of the task's transmission and execution times. $\Gamma_{t,s}^{trans}[n] = \frac{d_t[n]}{\mathcal{R}_{t,s}}$, $\Gamma_{t,s}^{exe}[n] = f_s[n]/f_s^{t,s}$, where $f_s^{t,s}$ denotes the frequency allocation by $\mathrm{SV_s}$ for the offloaded task. Consequently, total time for task completion is $\Gamma_{t,s}^{V2V}[n] = \Gamma_{t,s}^{trans}[n] + \Gamma_{t,s}^{exe}[n] + \Gamma_{t,s}^{wait}$, where $\Gamma_{t,s}^{wait}$ is the queueing time and total energy consumed is $\mathcal{E}_{t,s}^{V2V} = \kappa_t d_t f_s^{t,s}$.

V2I Offload: The data transfer rate from $\mathrm{TV_t}$ to $\mathrm{RSU_r}$ during time slot n is determined by $\mathcal{R}_{t,r}[n] = \mathcal{B}\log_2\left(1+\frac{\omega_t[n]h_{t,r}[n]}{\sigma^2}\right)$, where $\mathcal{B}$ stands for the bandwidth of the network, and $h_{t,r} = \beta_0(\lambda_{t,r}[n])^{-2}$ is used to express the channel gain between $\mathrm{TV_t}$ and $\mathrm{RSU_r}$, with β_0 indicating the base channel gain. The term σ^2 represents the noise power of the additive white Gaussian noise within the system. Furthermore, the duration required to transmit a task from $\mathrm{TV_t}$ to $\mathrm{RSU_r}$ in time slot n is calculated as $\Gamma_{t,r}[n] = \frac{d_t[n]}{\mathcal{R}_{t,r}[n]}$. Hence, the energy utilized for transmitting data from $\mathrm{TV_t}$ to $\mathrm{RSU_r}$ is determined as $\mathcal{E}_{t,r}[n] = \omega_t[n]\Gamma_{t,r}[n]$. As a result, the computation delay for tasks offloaded from $\mathrm{TV_t}$ to $\mathrm{RSU_r}$, along with the associated variability, can be approximated by $\overline{\Gamma}_{r,t} = d_t[n]c_t[n]/f_r[n]$ for the delay, and $\Delta\Gamma_{r,t}[n] = \frac{d_t[n]c_t[n]\widetilde{f}_r[n]}{f_r[n](f_r[n]-\widetilde{f}_r[n])}$ for the deviation. Therefore, the total computation delay encountered by $\mathrm{RSU_r}$ in processing the task offloaded from $\mathrm{TV_t}$ is expressed by $\Gamma_{r,t} = \overline{\Gamma}_{r,t} - \Delta\Gamma_{r,t}[n]$. Similarly, the energy expended in processing the task offloaded from $\mathrm{TV_t}$ at $\mathrm{RSU_r}$ is expressed as $\mathcal{E}_{r,t}[n] = \kappa_r(f_r[n] - \widetilde{f}_r[n])^2 c_t[n]d_t[n]$, where κ_r denotes the switching capacitance of the RSUs. With the allocation of a specific computing frequency f_r^t for processing the task transferred from $\mathrm{TV_t}$ to $\mathrm{RSU_r}$, the execution duration of the task is adjusted accordingly. Therefore, the total duration required for task completion, as well as the cumulative energy used to transfer the task from $\mathrm{TV_t}[n]$ to $\mathrm{RSU_r}$, are determined by, $\Gamma_{t,r}^{V2I}[n] = \Gamma_{t,r}[n] + \Gamma_{t,r}^{exe}[n] + \Gamma_{r,t}[n]$ and $\mathcal{E}t, r^{V2I} = \mathcal{E}t, r[n] + \mathcal{E}_{r,t}[n]$, respectively.

V2I via UAV Offload: In the context of accurately modeled DTs, UAVs receive real-time data from both vehicles and RSUs during each sub-slot. This enables the calculation of transmission rates, such as $\mathcal{R}t, u[n] = \mathcal{B}\log_2\left(1+\frac{\omega_t ht,u}{\sigma^2}\right)$, for the link from $\mathrm{TV_t}$ to $\mathrm{UAV_u}$ at time n. Here, $h_{t,u} = \beta_0(\lambda_{t,u}[n])^{-2}$ quantifies the LoS channel gain from $\mathrm{TV_t}$ to $\mathrm{UAV_u}$. Likewise, $\mathrm{RSU_r}$-$\mathrm{UAV_u}$ rates are derived using $\mathrm{RSU_r}$'s power ω_r. Analogously, $h_{r,u}$ and $h_{s,u}$ are LoS channels. The desired channel gain $h_{t,u,r}$, considering $\lambda_{t,u,r}$ as $\mathrm{TV_t}$-$\mathrm{RSU_r}$ distance via $\mathrm{UAV_u}$, is given as $h_{t,u,r} = \phi\beta_0\lambda_{t,u,r}^{-2}$, with $\phi \in (0,1)$ indicating a propagation attenuation factor. Thus is $\mathrm{UAV_u}$ to $\mathrm{RSU_r}$ transmission rate is $\mathcal{R}_{u,r}[n] = \mathcal{B}_1\log_2\left(1+\frac{\omega_u h_{u,r}}{\sigma^2}\right)$ where transmitting power of the UAV is denoted by ω_u, and the LoS connection from $\mathrm{UAV_u}$ to $\mathrm{RSU_r}$ is denoted as $h_{u,r}$. Consequently, the durations required to transmit tasks from $\mathrm{TV_t}$ to $\mathrm{UAV_u}$ and subsequently from $\mathrm{UAV_u}$ to $\mathrm{RSU_r}$ are expressed as $\Gamma_{t,u} = (d_t[n])/(\mathcal{R}_{t,u}[n])$, and $\Gamma_{u,r} = d_t[n])/(\mathcal{R}_{u,r}[n])$, respectively. Therefore, the energy consumed in these transmissions are computed as $\mathcal{E}_{t,u} = \omega_t[n]\Gamma_{t,u}[n]$ and $\mathcal{E}_{u,r} = \omega_u[n]\Gamma_{u,r}[n]$.

Therefore, the combined time and energy required for transferring the task from $\mathrm{TV_t}$ to $\mathrm{UAV_u}$ and then from $\mathrm{UAV_u}$ to $\mathrm{RSU_r}$ are denoted as $\Gamma_{t,u,r}[n] = \Gamma_{t,u}[n] + \Gamma_{u,r}[n] + \Gamma_{t,r}^{exe}[n]$ and $\mathcal{E}_{t,u,r}[n] = \mathcal{E}_{t,u}[n] + \mathcal{E}_{u,r}[n] + \mathcal{E}_{r,t}[n]$, respectively.

3 Hybrid Federated Learning Approach (HAFL)

The HAFL approach employs a synchronization method similar to that of AFL, focusing on cooperative updates via gradient and variance-centric aggregation, alongside a dataset segmentation strategy informed by variance levels. The metho- dology unfolds as follows:

1. **Vehicle Selection Condition.** Vehicles's stay time within UAV range, $\mathbb{T}_{u,t}^{stay}$, exceeds the sum of average training ($\mathbb{T}^{train}$) and prediction ($\mathbb{T}^{pred}$) times.
2. **Data Partitioning.** The UAV-aggregator distributes the dataset $\mathbb{D}$ into T distinct, non-intersecting subsets $\mathbb{D}_1, \ldots, \mathbb{D}_T$, with each subset corresponding to the local dataset of a TV.
3. **Local Model Training.** Training starts with servers updating local models using respective datasets. For example, $\mathrm{TV_t}$ trains $\mathbb{D}_t$ with parameters θ using stochastic gradient descent [1]. For each task vehicle TV_t, the gradient g_t is determined by $g_t = \nabla_\theta L(\rho_{t,i})$, where the loss function $L(\rho_{t,i})$ is defined by averaging the losses over TV_t's dataset: $L(\rho_{t,i}) = \frac{1}{|\upsilon_{u,i}|} \sum_{x \in \mathbb{D}t} l(\rho t, i; x)$. Here, the data x contributes to the loss calculation where $l(\rho_{t,i}; x) = (x - \hat{x})^2$. Furthermore, incorporating a regularization term with parameter p, the enhanced local loss function becomes $m(\rho_{t,i}) = L(\rho_{t,i}) + \frac{p}{2}|\rho_u - \rho_{t,i}|^2$.
4. **Regularized Local Loss Function.** To address stragglers, the method employs a decay coefficient to modify delayed local gradients, which are then combined with existing gradients. After a set number of iterations, the improved model is sent to the UAV. The federated learning process refines UAV_u's global objective function $\mathcal{O}(\rho_u)$ by merging local objectives from TVs, aiming for the optimal global model $\rho^* = \max_{\rho_u} \mathcal{O}(\rho_u)$.
5. **Delayed Local Gradient.** To further handle stragglers, the current gradient merges with a delayed counterpart using a decay coefficient. At time-slot n, the local gradient update is $\nabla\tau_{t,i} = \nabla m(\rho_{t,i}) + \iota \nabla g_t^d$, where ι is the decay rate and ∇g_t^d the gradient delay. The local learning rate, η_l^{rate}, adjusts to $\eta_l \max(1, \log(\mathbb{Q}))$ based on the initial rate η_l and queue size $\mathbb{Q}$. This results in the updated local model $\rho_{t,i+1} = \rho_u - \eta_l \nabla \tau_{t,i}$. The global objective $G(\rho_u)$ combines the losses from all TV_t weighted by their data sizes, guiding the overall model optimization.
6. **Variance Calculation.** Gradients from each TV are weighted during aggregat- ion according to their variances, giving more influence to TVs with higher gradient variances. The variance of the gradient for $\mathrm{TV_t}$ is calculated using $\mathrm{Var}(g_t) = E[(g_t - E(g_t))^2]$, with $E(g_t)$ representing the average gradient.
7. **Weighted aggregation.** The UAV performs the aggregation of gradients by calculating a weighted average, incorporating updates from TVs within each federated learning cycle. The weights w_t for the gradients from each TV in this aggregation process are determined using $w_t = \sum_i \frac{\mathrm{Var}(g_i)}{\mathrm{Var}(g_t)}$, adjusting for the variance of gradients across all TVs.

8. **Asynchronous aggregation.** The HAFL algorithm employs asynchronous aggregation for efficient global model updates, accommodating vehicle movement. It integrates TV updates into the global model via weighted gradient aggregation, optimizing accuracy while minimizing content. The weight for each TV's update, denoted as χ_t, incorporates factors like distance, coverage, and transmission time. The refreshed global model, denoted by $\mathcal{M}'_u$, is updated with a learning coefficient η_l, following $\mathcal{M}_u - \eta_l \sum_{t=1}^{T} \frac{ld_t}{ld} \chi_t \nabla \tau_t$. Here, $\frac{ld_t}{ld}$ signifies the share of total data ld corresponding to each $\mathrm{TV_t}$. This method enhances the effectiveness and acceleration of the global model's learning progression by finely tuning the updates [3].

4 Problem Formulation

Utility. The proposed framework prioritizes high-priority tasks with strict maximum delay tolerance. The utility of $\mathrm{TV_t}$ for executing tasks locally is given by

$$\mu_t^{loc}[n] = \begin{cases} \zeta - e^{-(\gamma_t[n] - \Gamma_t^{loc}[n])} - \vartheta_{\mathcal{E}}^{loc}, & \Gamma_t^{loc} \le \gamma_t, \\ 0, & \Gamma_t^{loc} > \gamma_t, \end{cases} \tag{4}$$

where $\vartheta_{\mathcal{E}}^{loc} = \vartheta^{loc} \mathcal{E}_t^{loc}[n]$ represents the energy cost, and ζ symbolizes a positive constant. If a local task fails its deadline, its assigned value is considered null. For small high-priority tasks offloaded to SVs, the utility is given by

$$\mu_t^{SV}[n] = \begin{cases} \mu_{t,com}^{SV}[n], & \Gamma_{t,s}^{V2V} \le \gamma_t, \\ -\hbar_s, & \Gamma_{t,s}^{V2V} > \gamma_p, \end{cases} \tag{5}$$

s.t. $\mu_{t,com}^{SV}[n] = \varkappa \left(\zeta - e^{(-\gamma_t[n] - \Gamma_{t,s}^{V2V}[n])} \right) - \vartheta_{SV} \mathcal{E}_{t,s}^{V2V}$. Similarly, the utility for RSU-offloaded tasks completed within the deadline is calculated as

$$\mu_t^{RSU}[n] = \begin{cases} \mu_{t,com}^{RSU}[n], & \Gamma_{RSU} \le \gamma_t, \\ -\hbar_c, & \Gamma_{RSU} > \gamma_t. \end{cases} \tag{6}$$

s.t. $\mu_{t,com}^{RSU}[n] = \varkappa \left(\zeta - e^{(-\gamma_t[n] - \Gamma_{RSU}[n])} \right) - \vartheta_{RSU} \mathcal{E}_{RSU}[n]$, $\Gamma_{RSU}[n]$ and $\mathcal{E}_{RSU}[n]$ depend on whether the task is offloaded directly from $\mathrm{TV_t}$ to $\mathrm{RSU_r}$ ($\Gamma_{RSU}[n] = \Gamma_{t,r}^{V2I}$, $\mathcal{E}_{RSU} = \mathcal{E}_{t,r}^{V2I}$) or via $\mathrm{UAV_u}$ relay ($\Gamma_{RSU}[n] = \Gamma_{t,r}^{V2I}$, $\mathcal{E}_{RSU} = \mathcal{E}_{t,u,r}$). If a task fails the deadline, the utility is assigned a penalty $-\hbar_c$.

Problem Statement. Binary indicators $\alpha_t[n] = \alpha_t^L, \alpha_t^{SV}, \alpha_t^{RSU}$ are utilized to identify computation strategies: $\alpha_t^L = 1$ denotes local processing, while $\alpha_t^{SV} = 1$ and $\alpha_t^{RSU} = 1$ signal task offloading to SVs and RSUs, respectively. The objective is to enhance total utility, defined in the optimization problem $\mathcal{P}1$:

$$\begin{aligned} (\mathcal{P}_1): \max_{\alpha_p} \quad & \alpha_t^L \mu^{loc} + \alpha_t^{SV} \mu_t^{SV} + \alpha_t^{RSU} \mu_t^{RSU} \\ \text{s.t.} \quad & C1: \alpha_t^L + \alpha_t^{SV} + \alpha_t^{RSU} \le 1, \\ & C2: \alpha_t^L \Gamma_t^{loc}[n] + \alpha_t^{SV} \Gamma_{t,s}^{SV} + \alpha_t^{RSU} \Gamma_{RSU} \le \gamma_T, \end{aligned} \tag{7}$$

where $C1$ guarantees that each task is confined to a unique computation method, while $C2$ verifies that the task's completion time does not exceed the allowable delay limit. Additionally, it is specified that a TV is restricted to offloading just one task within each N period.

5 Proposed Solution

For efficient resolution of the non-convex problem $\mathcal{P}_1$, it's divided into two sub-problems. $\mathcal{P}_2$ aims to minimize completion time and maximize utility for V2V offloaded tasks. $\mathcal{P}_3$ concentrates on optimizing utility for tasks offloaded to RSUs. The sub-problems are defined as

$$(\mathcal{P}_2): \min_{D_{t,s}} \quad \Gamma_{t,s}^{V2V} \quad \text{s.t.} \quad \Gamma_{t,s}^{V2V} \leq \gamma_t, \qquad (\mathcal{P}_3): \max_{\mathcal{E}_{t,r}^{V2I}} \quad \mu_t^{RSU} \quad \text{s.t.} \quad \Gamma_{t,r}^{V2I} \leq \gamma_t.$$

In Problem $\mathcal{P}2$, SV selection for $\Gamma_{t,s}^{V2V}$ is influenced by factors such as relative velocity, direction, distance, queuing time, and computation capacity, given task size similarity. Furthermore, in Problem $\mathcal{P}_3$, RSU selection aims to maximize μ_p^{RSU}, considering the distance, direction, transmission time, delay, energy, and resources. Notably, the local execution utility, set by $f_t^{loc}[n]$, remains constant. When $f_t^{loc}[n]$ exceeds or equals $\frac{4}{3}c_t d_t$, tasks of smaller size and higher priority are handled locally; if not, they are transferred to SVs, ensuring task distribution is governed by practicality and urgency.

Our solution, MARS, is a model-free, distributed, multi-agent DRL-based system for optimal resource allocation within DT-driven IoV. It uses a Markov Game (MG) with G agents: $G-1$ agents manage individual RSUs and one oversees V2V interactions. Agents handle specific vehicle sets per time slot, operating in continuous observation and action spaces without a central controller, using a partially observable MG and AFL for RSU agent training. Cooperative learning among independent agents ensures holistic optimization. The state space encompasses the attributes of DTs and UAV features, whereas the action space comprises selectable RSUs and SVs. MARS employs dual reward mechanisms: $\mathbb{R}1(s,a)$ tailored for RSU agents and $\mathbb{R}2(s,a)$ designed for V2V agents, with their calculations as follows:

$$\mathbb{R}1(s,a)[n] = \psi \mu^{RSU} t[n]\,, \tag{8}$$

$$\mathbb{R}2(s,a)[n] = \psi / \Gamma_{t,s}^{V2V}[n]\,. \tag{9}$$

Here, binary value $\psi = 0$ denotes task failure. Thus, each agent The MARS algorithm uses the Bellman equation to make action decisions for the agents as

$$Q_j(s,a)[n] = \mathbb{E}[\mathbb{R}j[n] + \varphi\ \mathbb{E}[Q_r(s,a)[n]]]\,, \tag{10}$$

such that j can be either r for RSUs or t for V2V mode. φ symbolizes the discount factor for future rewards, $\mathcal{R}j$, which includes both $\mathcal{R}1$ and $\mathcal{R}2$,

Algorithm 1. Cooperative MARS algorithm

Gather details of the incoming task.
if task=crucial **then**
    Local execution.
else if Task= smaller high priority **then**
    if $f_p^{loc} \geq \frac{4}{3} c_p d_p$ **then**
        Computation mode = local.
    else
        computation mode= SV.
    end if
    Collect data about nearby SVs
    if in-range SV==0 **then**
        computation mode= RSU.
    end if
else
    computation mode= RSU.
end if
Initialize current environment, B_r, B_t, and set of agents.
for each agent i ranging from 1 to G **do**
    Configure the actor and critic networks using the parameters $[\theta_r^{\mathcal{P}}, \theta_r^{\mathcal{Y}}]$ and $[\theta_t^{\mathcal{P}}, \theta_t^{\mathcal{Y}}]$.
    Establish corresponding target networks.
end for
for every episode **do**
    Begin with initial observations and set both $\mathbb{R}1$ and $\mathbb{R}2$ rewards to 0.
    for each step within the episode n **do**
        if agent operates under RSU **then**
            Determine action according to policy π_r.
            Compute $\mathbb{R}1$ as per (8), transition to new state (s').
            Refresh Q-function following (10) with $j = r$.
        else if agent functions in V2V context **then**
            Choose action based on policy π_t.
            Generate $\mathbb{R}2$ as per (9), move to new state (s').
            Revise Q-function in line with (10) setting $j = t$.
        end if
        for every agent i **do**
            if transition fits within B_j, $(j \in r, t)$ **then**
                Add transitions to the suitable replay buffers B_r or B_t.
                Calculate critic loss according to (11).
                Update the policy as per (12).
                Adjust target networks for $j \in (r, t)$ using

$$\theta_j^{\mathcal{P}'} \leftarrow \kappa\theta_j^{\mathcal{P}} + (1-\kappa)\theta_j^{\mathcal{P}'} \text{ and } \quad \theta_j^{\mathcal{Y}'} \leftarrow \kappa\theta_j^{\mathcal{Y}} + (1-\kappa)\theta_j^{\mathcal{Y}'} \ ,$$

            end if
        end for
    end for
end for
Repeat until convergence.

denotes the instant rewards. The anticipated future rewards for RSUs and V2V communicat- ions are represented by Q_r and Q_t, respectively, where $Q_r = \mathbb{E}[\mathcal{R}1 + \varphi\mathbb{E}[Q_r(s', a')]]$ and $Q_t = \mathbb{E}[\mathcal{R}2 + \varphi\mathbb{E}[Q_t(s', a')]]$. For the training phase, distinct replay buffers, B_r for RSUs and B_t for V2V, are employed. Thus loss function is estimated as:

$$\mathbb{L}(\theta_j^Q) = \mathbb{E}[((Q_j(s^i, a^i) - (\mathcal{R}j^i + \varphi Q'_j(s^{i'}, a^{i'})))^2], \tag{11}$$

where Q'_j signifies the target networks. The objective of the actor is to enhance the total expected reward by refining the policy optimization function as

$$\mathcal{J}(\theta_j^{\mathcal{P}}) = \mathbb{E}[Q_j(s^i, a)|a = \mathcal{P}_j(o^i)], \tag{12}$$

where $\mathcal{P}_j(.)$ defines the actor function that converts observations into actions, applicable for RSUs ($j = r$) and V2V interactions ($j = t$). **Algorithm** 1 provides a concise overview of the MARS algorithm.

6 Simulation Results

The effectiveness of the MARS algorithm within the envisioned DT-supported IoV framework is evaluated in our study. For this purpose, we create a simulated environment, producing required data in alignment with prevailing system condit- ions. We leverage the capabilities of the OpenAI Gym toolkit for data generation. The speeds of vehicles are assigned following a normal distribution, adhering to limits set in the designated table (refer to Table 1). For the AFL model's training phase, a dataset comprising 51,000 instances is utilized, whereas the evaluation phase involves a set of 9,000 instances. Our simulation encompasses a scenario with a vehicle density ranging from 20 to 70 units per square kilometer. Additional details regarding the parameters, including the learning rate (denoted as LR) and the structure of hidden layers (denoted as HL), are systematically outlined in Table 1.

To illustrate the effectiveness of our proposed framework, we conduct comparative analyses with a variety of established methodologies, namely: (i) Multi-agent-DDPG (MA-DDPG), (ii) SFL-based MARS (SFL-MARS), (iii) SFL-based single-agent DDPG (SA-DDPG), and (iv) Random approach.

SFL requires the UAV to wait for all vehicle models before updating globally. SA-DDPG operates with a central controller and a single agent for RSU allocation. The random method assigns tasks based on current computation capacities.

1. Mean Utility (MU): Fig. 2 shows the variation of TVs' mean utility (MU) with task data sizes. It reveals that larger tasks lead to lower MU, attributed to increased resource demands, extended processing time, and heightened energy usage. The proposed AFL-inspired MARS algorithm surpasses established bench- marks such as AFL-enhanced MA-DDPG, SFL-enhanced MARS, both AFL and SFL variations of SA-DDPG, and the Random approach, securing notable gains in utility. SFL-MARS and SA-DDPG

Table 1. Simulation parameters

Param.	Values	Param.	Values
LR_{actor}	0.001	η_l	0.01
LR_{critic}	0.002	R	5
HL_{actor}	[1000, 1000]	$\mathcal{F}^{RSU}$	[7, 8, 9]GHz
HL_{critic}	[800, 800]	$\mathcal{F}^{SV}$	$[0-5]$GHz
$\mathcal{F}^{loc}$	$[0.5-3]$GHz	κ	0.1
$[\omega_t, \omega_r, \omega_u]$	[10, 25, 45]dBm	$\mathcal{L}$	$30-50$ms
optimizer	Adam	T	$4-8$
max. velocity	50km/hr	H	$200m$
Channel gain/m	-50dB	U	1
max. vehicle range	80m	batch	32
max. RSU range	300m	p	0.0001
max. no. of tasks per time period	30	$\mathcal{D}$	[300 − 1200]kB
activation func.	ReLu	ι	0.001
max. distance TV-SV	50m	φ	0.99

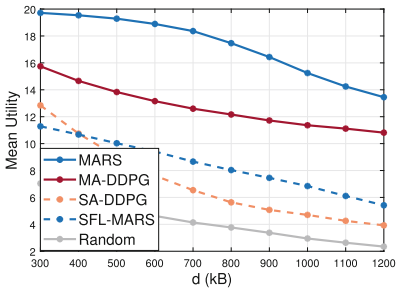

Fig. 2. MU vs d.

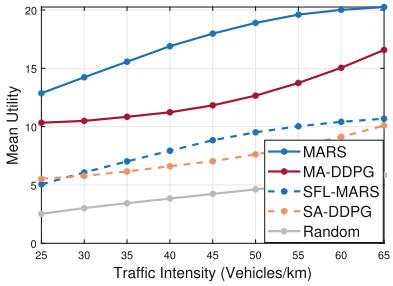

Fig. 3. MU vs traffic.

exhibit unique performance trajectories, with SFL-MARS showcasing superior utility for extensive task data sizes, thanks to its efficient distribution of workload across numerous agents.

Likewise, Fig. 3 illustrates the MU performance of TVs across different levels of traffic density, where the MARS algorithm reliably exceeds the performance of alternative methods. MA-DDPG outshines SFL-related strategies, and SA-DDPG exhibits marginally improved results in scenarios with minimal traffic. Yet, with the escalation of traffic density, SFL-MARS showcases enhanced utility, in contrast to SA-DDPG. The Random algorithm performs unsatisfactorily in both Figs. 2 and 3.

2. Task completion factor (TCF): Fig. 4 provides a comparative analysis of task completion rates for different data sizes, contrasting our algorithm with established benchmark approaches. TCF denotes a ratio that measures

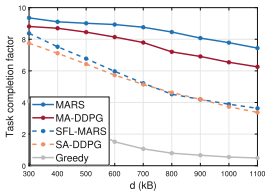

Fig. 4. TCF vs d.

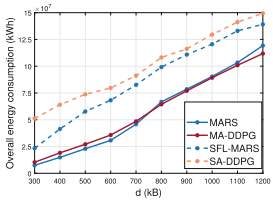

Fig. 5. $\mathcal{E}_{con}$ vs d.

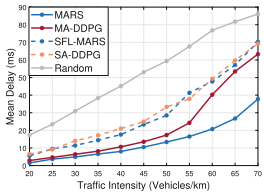

Fig. 6. MD vs traffic.

the number of completed tasks out of every 10 task, which is used to compare the algorithms. It's noticeable that as the data size increases, the task completion rate decreases across all considered algorithms. The proposed algorithm gives the highest TCF, followed by the MA-DDPG algorithm, indicating that the HAFL approach enhan- ces the resource allocation quality in the framework. On the other hand, the SFL-based methods demonstrate lower task completion rates. The behavior of SFL-MARS and SA-DDPG is almost identical, with SFL-MARS performing slightly better than SA-DDPG. The random approach achieves a very low task completion rate.

3. Overall energy consumption ($\mathcal{E}_{con}$)**:** Fig. 5 depicts the relationship between overall energy consumption and task sizes ranging from 300 to 1120 kB. As task sizes increase, energy consumption also grows due to prolonged processing and completion durations. This figure contrasts the energy efficiency of the MARS algorithm with MA-DDPG utilizing HAFL and SA-DDPG utilizing SFL. Results show that both HAFL-driven MARS and MA-DDPG algorithms exhibit similar energy consumption, whereas SFL-driven MARS consumes slightly more energy, and SA-DDPG shows the highest energy consumption. This difference is attributed to the frequent interactions required by SFL with multiple clients, leading to increased energy usage and longer training times. Additionally, SA-DDPG imposes greater computational demands on the agent, resulting in higher energy consumption and extended processing times.

4. Mean Delay (MD): Fig. 6 demonstrates that the mean system delay (MD) escalates as traffic density increases, attributed to greater congestion that results in lengthier transmission and processing times. In scenarios of varying traffic densities, the MARS algorithm exhibits superior performance in reducing MD compared to competing algorithms. Under condi- tions of low traffic, MA-DDPG's performance is on par with MARS; however, its delay experiences a substantial increase with the rise in traffic density. SFL-influenced strategies, including SFL-MARS and SA-DDPG, display comparable patterns, with SFL-MARS generally achieving a marginally better MD. The Random approach experiences considerably greater delays across all traffic conditions.

7 Conclusion

In this research, we introduce a method to enhance resource allocation in DT-supported UAV-assisted IoV networks. By integrating HAFL with multi-agent DRL, our approach focuses on improving task completion rates, lowering energy usage, and reducing system delays. We presented a DT framework with three types of DTs and three modes for computing tasks, enabling V2V and V2I offloading. Additionally, we introduced MARS, a multi-agent DRL algorithm tailored for resource allocation optimization in DT-empowered IoV networks, trained using HAFL. This hybrid AFL technique is aimed at better resource manage- ment to improve the system's utility. Our evaluations, conducted through simula- tions, indicate that MARS outperforms both conventional DRL and SFL-based MARS in task completion, energy efficiency, and delay minimization for UAV-assisted IoV networks.

References

1. Bian, J., Wang, L., Yang, K., Shen, C., Xu, J.: Accelerating hybrid federated learning convergence under partial participation. arXiv preprint arXiv:2304.05397 (2023)
2. Do-Duy, T., Van Huynh, D., Dobre, O.A., Canberk, B., Duong, T.Q.: Digital twin-aided intelligent offloading with edge selection in mobile edge computing. IEEE Wireless Commun. Lett. **11**(4), 806–810 (2022). https://doi.org/10.1109/LWC.2022.3146207
3. Hazarika, B., Singh, K.: AFL-DMAAC: Integrated resource management and cooperative caching for URLLC-IoV networks. IEEE Trans. Intell. Veh. pp. 1–16 (2023). https://doi.org/10.1109/TIV.2023.3303932
4. Hazarika, B., Singh, K., Biswas, S., Li, C.P.: DRL-based resource allocation for computation offloading in IoV networks. IEEE Trans. Industr. Inform. **18**(11), 8027–8038 (2022)
5. Hazarika, B., Singh, K., Biswas, S., Mumtaz, S., Li, C.P.: Multi-agent DRL-based task offloading in multiple RIS-aided IoV networks. IEEE Trans. Veh. Technol. (2023)
6. Hazarika, B., Singh, K., Li, C.P., Schmeink, A., Tsang, K.F.: RADiT: Resource allocation in digital twin-driven UAV-aided internet of vehicle networks. IEEE J. Sel. Areas Commun. (2023)
7. Wang, S.H., Tu, C.H., Juang, J.C.: Automatic traffic modelling for creating digital twins to facilitate autonomous vehicle development. Connect. Sci. **34**(1), 1018–1037 (2022)
8. Wu, Q., Zhao, Y., Fan, Q., Fan, P., Wang, J., Zhang, C.: Mobility-aware cooperative caching in vehicular edge computing based on asynchronous federated and deep reinforcement learning. IEEE J. Sel. Top. Signal Process. **17**(1), 66–81 (2023). https://doi.org/10.1109/JSTSP.2022.3221271
9. Yang, H., Zhao, J., Xiong, Z., Lam, K.Y., Sun, S., Xiao, L.: Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management. IEEE J. Sel. Top. Signal Process. **39**(10), 3144–3159 (2021)