The Journal of Supercomputing
https://doi.org/10.1007/s11227-022-04911-8

A federated multi-agent deep reinforcement learning for vehicular fog computing

Balawal Shabir[1] · Anis U. Rahman[1] · Asad Waqar Malik[1] · Rajkumar Buyya[2] · Muazzam A. Khan[3]

Accepted: 21 October 2022

Abstract

Vehicular fog computing is an emerging paradigm for delay-sensitive computations. In this highly dynamic resource-sharing environment, optimal offloading decision for effective resource utilization is a challenging task. In recent years, deep reinforcement learning has emerged as an effective approach for dealing with resource allocation problems because of its self-adapting nature in a large state space scenario. However, due to high mobility and rapid changes in the network topology cause fluctuating task arrival rate. Similarly, the data sharing between the vehicles and the fog nodes raises a variety of security and privacy concerns. Therefore, the proposed system is based on local and global model training approaches. In this paper, we propose a federated multi-agent deep reinforcement learning solution that efficiently learns task-offloading decisions at multiple tiers i.e. locally and globally. The proposed work results in fast convergence due to its collaborative learning model among vehicles and fog servers. The local model runs at the vehicular nodes, and the global model runs at the fog servers. To reduce network overhead, the models are learned locally; thus, limited information is shared across the network this reduces the communication overhead and improves the privacy of the agents. The proposed system is compared with the greedy and stochastic approaches in terms of residence times, cost, delivery rate, and utilization ratio. We observed that the proposed approach has significantly reduced the task residence time, end-to-end delay and overall system cost.

Keywords Deep reinforcement learning · Federated learning · Computation offloading · A3C · Advantage function · Residence time

✉ Asad Waqar Malik
asad.malik@seecs.edu.pk

Extended author information available on the last page of the article

Published online: 31 October 2022

1 Introduction

With the rapid growth in Internet-of-Things (IoT), V2X communication has become more prevalent for efficient communication and computation [1]. The vehicular agents communicate with other devices, share the underutilized computation resources to support the delay-sensitive applications. Particularly, the vehicular network exhibits highly dynamic and nonlinear behavior and decision making in vehicular systems is challenging due to the continuously varying state of the resource sharing environment [2]. The time-critical nature of vehicular applications, high mobility and rapidly varying queue state makes task offloading a challenging task [3]. Aforementioned issues can be mitigated by introducing machine learning based solutions. One traditional solution uses game theory and reinforcement learning for decision optimization by collecting perpetual feedback from the environment [4]. Such solutions fail to capture the large state space that corresponds to the frequently varying queue states of the nodes. Comparatively advanced solutions employ deep reinforcement learning for decision optimization but they are mostly compute-intensive and yield slower speed of convergence [5]

To achieve faster convergence, policy optimization-based solutions are adopted that can approximate the task offloading strategy in a large state space but converge to local minima due to uncertainty of the environment, limited information sharing and centralized architecture There is a limited knowledge sharing among vehicular agents and presented solutions are centralized in nature as they collect knowledge from the central fog servers [6]. Therefore, a new paradigm of distributed learning named federated learning can play an indispensable role to achieve improved decision making and fast convergence [7, 8]. Federated learning accomplishes fast convergence by learning locally on device and also attains low communication overhead by transmitting less data across the resource sharing environment.

The Federated learning (FL) is a type of distributed learning suitable for situations where training locally has an advantage [9]. The learning approach leverages the modern smart vehicles with growing computational and storage capabilities [10]. That is, training a model with a large set of input parameters becomes unsuitable when storing and processing at a central data server with limited storage and computing resources. This centralized approach incurs significantly high communication costs when transmitting raw input data across the network. Most importantly, the data is often privacy sensitive, which needs to be kept locally rather than communicated over the network [11]. The objective of FL is to collaboratively come up with a global prediction model, sharing learned model parameters while keeping the training data local. Generally, in data-parallel FL approach [12], the learning starts off with vehicles collecting data and using it to train independent local models. This is followed by an aggregator combining all learned local model parameters to produce a global model, which is used to retrain an updated local model. The iterative learning process continues until the learning algorithm reaches convergence. There are a few limitations of FL, one that the approach is not fully distributed because it relies on a central aggregator,

even though full decentralization leads to loss of learning model efficiency. Another limitation is that the approach is prone to malicious vehicles deliberately altering local training data or selfish vehicles acting non-cooperatively when sharing local models [13]. Nevertheless, the approach is feasible for providing more robust and time-efficient innovative vehicular services, especially in the context of autonomous vehicles.

Reinforcement learning (RL) intelligently learns the optimal policy to maximize reward based on perpetual feedback from the environment [14]. This is analogous to humans interacting with their outer environment so that to use the interactions to understand it without example [15]. However, the time required to fully explore the complex environment and finally converge to an optimal policy makes traditional RL infeasible. This is especially challenging when scaling RL for large and dynamic environments with massive and continuous state-action space. To address such large-scale spaces, deep reinforcement learning (DRL) is introduced to provide better generalization and adaptation to unknown environments [16]. Here, an RL agent interacts with the complex environment and the deep network to approximate the value function or policy, accelerating the agent's capability to learn an optimal policy for future actions. The usefulness of DRL has been demonstrated for solving problems including control [17], resource management [18], games [19], robotics [20] and autonomous driving [21].

Notably, RL is effective when performing sequential decision-making under uncertainty [22]. This is the case in highly dynamic mobility scenarios where offloading constraints are more implicit and diverse in comparison to traditional optimization problems with explicit rules [5]. For instance, vehicle-to-everything (V2X) supports various services with different service requirements but with its unique and challenging dynamics. Moreover, the explosive growth in vehicular technology adoption is another reason for interest in RL-based offloading decision models. In general, these models involve training rewards that correlate with an objective, and eventually reaching an optimal strategy. Interestingly, this training is possible using distributed algorithms, meaning the strategy can be devised by working collaboratively in a distributed manner, improving system-level performance based on local information. In this work, we propose a DRL-based federated offloading framework that achieves efficient task offloading decisions by collective learning the tasks offloading decisions at multiple tiers. The framework provides an extended learning capabilities through knowledge sharing among the vehicles and the fog servers. The main contributions of the work are:

- Explore multi-agent V2X scenario where the agents interact with the vehicular environment, and collaboratively understand its dynamics for better task offloading decisions.
- Design a tiered federated offloading system that decentralizes edge intelligence by learning offloading decisions at multiple tiers with multiple distributed learning environments. This improves the usage of communication channel and data privacy with limited information sharing across tiers.
- Implement a two-tiered offloading system using asynchronous advantage actor-critic (A3C) algorithm, a hybrid reinforcement learning algorithm. The local

models are trained on vehicular agents while the global model is trained at the fog-servers. A3C captures the bigger part of the state space by running multiple isolated learning environments. This efficient exploration of the state space and multi-tier learning achieves faster convergence with an improved offloading decision knowledge.

The rest of the paper is organized as follows: Sect. 2 discusses the related work. Section 3 covers the system model. Section 4 we provided the conceptualization of our model and formulated the problem. Section 5 covered the proposed framework with the result evaluation and discussions in Sect. 6. Finally, the conclusion is presented in Sect. 7 (Table 1).

Table 1 The Notations Definitions

P_{t}	Transmission power
$\mathcal{V}$	Set of vehicles
$\mathcal{P}$	Parked vehicles
$\mathcal{F}$	Fog servers
d	Distance
l	Path loss
C	Channel capacity
$\mathbf{B}$	Bandwidth
N_0	Channel noise
s_j	Task size
c_j	Computing requirements
λ	Arrival rate
μ	Service rate
s_i	Input size
Z_i	Residence time
$\mathcal{Q}$	Queue length
Z_i	Residence time
f_i	Clock frequency
D_i^1	Transmission delay
ψ_i	Frequency reuse cost
ψ_2	Leased computation cost
D_i^2	Computation delay
D_i^3	Queuing delay
R_i	Agent reward
r_i	Action reward
T	Total timesteps
$\mathcal{M}_i$	Total received task

2 Related work

This section covers the recent contributions made to efficiently utilize computing nodes available across a vehicular network.

2.1 Conventional task offloading

Considers the vehicle speed and its direction to determine the task offloading decision. That is, vehicles moving in the same directions become potential candidates for the task offloading. For instance, in [23], mobility-aware and location-based task offloading schemes are presented for individual and cooperative fog-servers. The work effectively uses communication and computation resources while meeting task deadlines. This approach only considers device mobility for resource selection without considering its queue state. However, in [24], a context-aware opportunistic offloading based on vehicle direction, speed, position and queue state is proposed to improve resource utilization in a vehicle-to-vehicle (V2V) resource-sharing network. In [25], an adaptive task offloading technique based on multi-armed bandit theory is proposed that relies on offloading delay of fog nodes. To reduce the workload on the nearby fog-servers, Zhang et al. [26] propose a game-theoretic approach for load balancing in a FiWi-enabled fog-computing network. The solution relies on the concept of software-defined networks for centralized network management. In [27], the available resources are pooled and shared in the form of an incomplete information game strategy for task offloading. It selects resources based on a greedy-pruning algorithm to minimize energy consumption and maximizing resource utility. Similarly, in [28], a genetic algorithm is used to optimize resource allocation from the pool. Notably, such meta-heuristics techniques depend on centralized network management, and they are more focused on associated device parameters for resource selection. These techniques perform well for discrete optimization problems but degrades performance with increasing requests and fluctuating queue states. Furthermore, AI-based task offloading captures large state spaces and adapts to the dynamic vehicular environment. For instance, in [29], a prediction-based model is used to minimize execution delay. In [30], multi-model multi-task offloading is proposed where multiple UAVs are deployed to efficiently compute high-requirement tasks using ensemble learning. In [31], an auction-based technique for fog-server allocation to incoming requests from mobile nodes. The technique uses a deep neural network (DNN) model to minimize energy utilization and delay constraints of mobile nodes. Similarly, in [32], a DNN-based task scheduling and computation offloading scheme is implemented for edge computing. The evaluation results demonstrate improved resource scheduling. Notably, such DNN-based methods are not suitable for rapidly changing vehicular environment with large state space and limited information sharing, impact their decision-making capabilities.

2.2 Reinforcement learning-based offloading

More recent attempts like in [33] present an intelligent task offloading scheme based on deep Q-learning where the information and network data throughput is managed through a centralized controller. Similarly, in [34], a computing resource allocation scheme based on deep Q-network is presented to alleviate constraints of traditional Q-learning. It considers vehicle speeds and their computing powers to determine an effective decision model for task offloading. However, Q-learning becomes inefficient for large state space scenarios with slow convergence due to frequent policy explorations and relevant computations. In [6], a service offloading framework performs online learning at the vehicles, which is later used to learn an optimal offloading policy at the fog device. In [7], a 'when and where to schedule' model formulates a stochastic resource optimization problem using DRL. The model ingests a characteristic feature set for task queue states, which is used to approximate a task assignment strategy using DNN. A relevant study in [35] proposes a priority-aware DRL solution with incentives for vehicles to share their idle computational resources. The solution defines a dynamic pricing strategy based on task priority, node mobility and resource availability. Similarly, in [36], a DRL-based offloading strategy optimizes a joint objective function incorporating task queue state, vehicular mobility and task dependency. The strategy optimizes the function by minimizing energy consumption and task offloading delay. In [35] presents an interesting work on the task offloading paradigm based on deep reinforcement learning. The paper uses a model free deep reinforcement learning based on the SAC (soft actor critic) algorithms which maximizes the entropy and the expected utility of the task. However, the presented work is centralized in nature where the task offloading decisions are only learned on the base station. Unlike our proposed solution, where two different models are learned at the vehicles and the base stations respectively, the presented work is not federated in nature and learns the task offloading decisions only at the base stations. Notably, the DRL solutions converge to a local optimum due to the inherent uncertainties of the dynamic environment and limited knowledge sharing in continuous state space scenarios. Interestingly, distributed and federated DRL may well serve this purpose by ensuring a diverse training environment while resulting in faster convergence to global optimum (Table 2).

2.3 Summary of literature

Most of the proposed work considers the context of the neighboring vehicles to optimize the offloading decision. Conventional offloading techniques are suitable for the environments with little or no mobility and optimize a few parameters, limiting their decision-making capabilities. These solutions are not able to predict the environment state that corresponds to the optimal offloading decision. Recent work uses deep reinforcement learning but it slowly converges to the optimal solution. We propose a federated and multi-tier deep reinforcement learning that quickly converges to an optimal solution resulting in an improved resource utilization. The federated

Table 2 Summary of the contributions of this paper and comparison with other relevant reported works

Authors (year)	Architecture	Technique	Fast convergence	Collaborative learning	Load balancing	Multi-agent	Communication efficiency	Privacy
Rahman et al. [24]	Distributed	Context-aware	X	✓	Fog consortium	X	X	X
Chen et al. [32]	Centralized	Greedy	X	X	Fog consortium	X	X	X
H Guo et al. [33]	Centralized	Deep Q-Learning	X	X	Fog consortium	X	X	X
Qi Qi et al. [6]	Distributed	A3C	✓	✓	Fog consortium	X	✓	X
W. Zhan et al. [7]	Centralized	Policy Gradient(PPO)	✓	✓	Fog consortium	X	X	X
J. Shi et al. [35]	Centralized	Soft Actor Critic(Gradient)	✓	X	Fog consortium	X	✓	X
B. Lin et al. [36]	Distributed	SA-DQN	✓	X	Fog consortium	X	X	X
Proposed	Distributed	Actor-critic	✓	✓	Fog consortium	✓	✓	✓

learning corresponds to faster convergence, minimizes the objective function achieving low residence time and improved quality of experience for vehicular services. Proposed method is suitable for the scenarios where there is limited data sharing among the nodes and a faster convergence is required. It is suitable for the problems where multiple distributed agents work in parallel to achieve a common goal [37]. Deep reinforcement learning has proved to be an effective tool for the variety of applications including intelligent transportation systems, smart grids, block-chain empowered IoT, mobile crowd sensing, industrial IoT applications, etc.

We further assumed that due to the highly mobile vehicular environment, the vehicles may not be connected with the base station. As discussed, the optimization problem presented in our case is solved at the lower tier where vehicles learn the task offloading model in a distributed fashion. As we have employed the A3C algorithm which is inherently distributed in nature by running multiple simulation environments in parallel. Similarly, the vehicular agents are also interacting in their respective simulation environments. Each vehicle interacts with the environment, collects experience in an online fashion and optimizes its offloading decisions. As discussed in the algorithm the upper tier is responsible for the aggregation of the parameter received from the lower tier and return the updates back. The upper tier is helping the agents to come up with a more diversified training experience by accumulating updates for all the vehicles in a simulation scene.

3 System model

We consider a resource-sharing vehicular environment comprising a set of moving vehicles denoted as $\mathcal{V}$, a set of parked vehicles $\mathcal{P}$, and a set of fog-servers $\mathcal{F}$ deployed at different locations across the vehicular network. Note that only the moving vehicles generate tasks with varying computational requirements, they are either computed locally or offloaded to neighboring nodes within the vehicular environment.

3.1 Transmission model

Let d denote the distance between devices, the path loss l for wireless channel is given as [38],

$$l = a + \beta \log_{10}(d) + b \tag{1}$$

where a and b are the initial offset and shadow fading effect, respectively, with β as a path attenuation index. So, the upper bound at the communication channel is given as,

$$C = \mathbf{B} \log_2 \left(1 + \frac{P_t \cdot l}{N_0}\right) \tag{2}$$

where C is the channel capacity for the radio channel with bandwidth $\mathbf{B}$. Transmission power of the vehicular nodes is P_t, whereas channel noise is denoted as N_0. Note that we assume a quasistatic and time-invariant wireless channel [39]. It means that all nodes experience a homogeneous channel state with same transmission power and interference level. Recall, the channel capacity is included in terms of the transmission delay, an important part of the end-to-end delay which we want to reduce. For channel capacity, we model consider distance, pathloss and adapts based on the changing values of the transmission delay to learn better tasks offloading preference.

3.2 Task model

Different vehicular applications generate tasks that have the same priority without any interdependency among tasks. Each arriving task $j \in J$, (s_j, c_j) is represented by two tuples where s_j denotes the task size in MBs and c_j is the computing requirements in Mbits.

3.3 Queuing model

We consider an infinite population multi-server FIFO queuing model with the system workload as the Poisson process and an arrival rate λ. There are $\mathcal{N}$ mobile nodes communicating with the exponentially distributed service rate μ. The total system capacity is denoted as $\forall(\mathcal{V}, \mathcal{F}) = \sum_{i=0}^{T} \mathcal{K} \cdot (c \cdot s_i)$, where $\mathcal{K}$ denotes the size of calling population, c denotes the number of parallel compute nodes and the s_i denotes the input data size. The residence time Z_i of an agent i is denoted as,

$$Z_i = \sum_{x \in \mathcal{Q}} \frac{c_x}{f_i} \tag{3}$$

where, $\sum_{x \in Q} c_x$ denotes the queue length of queue $\mathcal{Q}$ whereas f_i is the clock frequency.

3.4 Task execution model

While offloading task, a portion of the bandwidth is occupied by the transmitting node. The total transmission delay $D_i^{\downarrow}$ for vehicle $i \in \mathcal{V}$ is denoted as,

$$D_i^{\downarrow} = \psi_i \mathbf{B} \cdot (D_i^{\uparrow} + D_i^{\downarrow}) \tag{4}$$

Here, ψ_i represents the cost of the frequency reuse and $D_i^{\uparrow}$ and down-link $D_i^{\downarrow}$ represents the up-link and down-link transmission delays. The up-link communication delay of a task j is denoted as, $D_i^{\uparrow} = \sum_{j \in J} \mathbf{1}_J \cdot (\frac{s_j}{C_i})$. Similarly, the indicator random variable $\mathbf{1}_J(\cdot)$ is whether or not a task is offloaded. We ignored the output delay due to its small size [39]. Furthermore, the computation delay D_i^2 for a vehicle i is represented as,

$$D_i^2 = (\psi_2 f_i) \cdot D_i^* \tag{5}$$

where, ψ_2 is unit lease cost of the computation resource and f_i is the clock frequency of the computing node. Computation time $D_i^* = \sum_{j \in J} \frac{c_j}{f_i}$ represents the time to service a task. Similarly, queuing delay is the waiting time of the task in the queue before it gets executed. So, queuing delay D_i^3 is represented as,

$$D_i^3 = (\psi_2 f_i) \cdot \sum_{j=1}^{J} Z_j \tag{6}$$

where Z_j represents the residence time which indicates the time for which task j stays in the system queue and gets executed. ψ_2 is the unit lease cost of the computation resource for queue management. Similarly, the total service time D_i is the summation of all delays,

$$D_i = D_i^1 + D_i^2 + D_i^3 \tag{7}$$

3.5 Offloading model

As mentioned earlier, the vehicles generate tasks that are either executed locally or offloaded across the resource-sharing vehicular network. This decision is based on computation requirements of the task and available resources including fog-servers and parked vehicles. Specifically, the decision model at a source vehicle i for a task j considers an extensive feature set when deciding to offload a task to neighboring nodes, given as,

$$\delta_i = \left\{(0,1) : \delta = \mathbf{f}(x_1, x_2, x_3, \cdots, x_n)\right\}, \forall j \in J \tag{8}$$

where $(x_1, x_2, x_3, \cdots, x_n)$ is the feature set corresponding to the current state of the vehicle i. The function $\mathbf{f}(\cdot)$ for an agent i is the mapping from the agent's state to an offloading mode δ_i learned by the task offloading algorithm.

4 Problem formulation

Vehicular agents interact with the vehicular environment to improve their offloading decisions based on prior experience. Consider an agent i that selects an offloading action a_i based on its current state s_i, subsequently, a reward r_i get into the next state s_i'. This interaction of the agent with the environment can be represented in the following form:

4.1 State space

The current state of the vehicle represents the availability of the computing resources at the current instance. The computing demand varies drastically in

vehicular systems so vehicles must fulfil the computing requirements-based computing requirements and the availability of the computing resources. Our solution considers the variety of the parameters related to the tasks and the vehicles. We have considered a holistic model to devise the state of the system considering the task computing requirement and computing capabilities of the nodes. We have also considered the data rates and the transmission delays of the vehicles that control the flow rate of the tasks. Generally, a state space for a vehicular agent corresponds to the set of relevant parameters that effectively capture the dynamics of the surrounding vehicular environment. For this purpose, we define a state profile $\mathcal{S}_i$ for an agent i with a job j as $\mathcal{S}_i = \{\mathcal{Z}_i, \Delta_i, \mathcal{V}_i, \mathcal{W}_i, f_i, (s_j, c_j)\}$.

- $\mathcal{Z}_i$: is a set of residence times of available resources including local, fog-servers, moving and parked vehicles.
- Δ_i: is a distance set from the task source vehicle to neighboring resources including fog-servers, and moving and parked vehicles.
- $\mathcal{V}_i$: is a set of relative speed to neighboring resources including fog-servers, and moving and parked vehicles.
- $\mathcal{W}_i$: is a set of offloading modes of available resources including local, fog-servers, and moving and parked vehicles.
- f_i: corresponds to task source agent's computing capacity.
- (s_j, c_j): represents the task attributes including the input data size and required frequency cycles.

Notably, the performance metrics should be dynamically measured for the changing vehicular environment. As it is difficult to completely capture the dynamics of the networks due to rapidly changing topology and a very large state space. Hence our state creation module dynamically generates the environment state in real time and uses that experience to select the tasks offloading action at the current instance.

4.2 Action space

In a resource-sharing vehicular network, an offloading action space describes the set comprising the available actions for task execution. That is, this space corresponds to computational resources available locally, at fog-servers, and moving or parked vehicles. Consider for a vehicle agent i with task j, the action space $\mathcal{A}_i$ is given as,

$$\mathcal{A}_i = \{a_1, a_2, a_3, a_4\} \tag{9}$$

where a_1 queues the task locally for execution; a_2 offloads the task to a neighboring moving vehicle; a_3 offloads the task to a nearby fog-server; and a_4 offloads the task to a nearby parked vehicle.

4.3 Reward function

A reward function guides a learning algorithm to achieve its objective function. Here, we define one based on residence times, that is it maximizes the reward at

every subsequent iteration by selecting an offloading action among the available action space to minimize the residence time. Say for an action a_i taken by a vehicle agent i with a task execution at $k \in \{i, \mathcal{X}_i\}$, the reward r_i is represented as $r_i = -\left(\frac{1}{\lambda}\right)Z_k$ where, λ is the task arrival rate and Z_k is the residence time of the selected destination node k. Hence, the total accumulated reward R_i for an agent i is defined as,

$$R_i = \sum_{t=0}^{T} \frac{r_i(t) - R_i}{\mathcal{M}_i} \tag{10}$$

where r_i is the received reward against an action a_i at timestep t in a simulation of total T timesteps, and $\mathcal{M}_i$ is the total number of tasks received by the agent i.

To conclude, the agents start the learning process by receiving the state vector as an input. The local and global model sharing helps the agents to better perceive the environment state and results in improved decision making. Initially the decisions are taken randomly and based on the rewards received against an action, the agents improve their action. The reward is the function of the residence time and selection of nodes with the lower residence time produce the higher value of the rewards. So, based on the availability of the vehicles in the surrounding area, the vehicular agents prefer the action with the higher reward which results in lower residence time.

4.4 Multi-agent A3C algorithm

Actor-critic networks is a policy gradient algorithm that parameterizes the policy $\pi(a|s;\theta)$ by updating the parameter θ. These algorithms are inherently different from traditional Q-learning and deep Q-nets (DQN) where the best action is determined based on the current state, meaning the action with the maximum Q value. In contrast, policy gradients directly learn the policy π or state to action mapping suitable for the environment with large state space. In general the actor-critic network comprises of two states, the actor is responsible for taking action according to policy whereas the critic appraises this action to improve the actor's decision. This improvement is represented by a reduction in variance or advantage, computed as $R_t - b_t(s_t)$ by subtracting the cumulative return R_t from a baseline function. The baseline is usually referred to as the learned value estimate $V^{\pi}(s_t)$.

Considering a multi-agent deep reinforcement setting, we have a group of $\mathcal{N}$ agents interacting within the vehicular environment. This forms a basic Markov decision process (MDP) with tuples representing agents such as (s, a, r, s'). Here, s is the current state of the agent, a is the action chosen by the agent, r is the reward associated with the action taken, and s' is the next state of the agent. Say at an instant t, the $\mathcal{N}$ agents interact with the environment, taking independent actions with a transition probability p and getting rewarded for them while jumping to their next states s'. The goal of an agent is to learn a state to action mapping $\pi(s) \rightarrow a$ that maximizes the expected discounted future reward, represented as $V^{\pi}(s_t) = \mathbb{E}(R_t|(s_t, a_i))$. Here, R_t is the cumulative discounted future return computed as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+1}$ with a discount factor $\gamma \in (0, 1]$.

In practice, A3C algorithm runs concurrent threads for model training, meaning it improves its training efficiency by running multiple parallel threads with diverse and improved learning experience. Here, multiple agents simultaneously explore separate parts of vehicular state space. Due to multiple asynchronous interaction the updates are uncorrelated, which explores the larger part of the state space in less time. At a time instant t, the algorithm calculates an advantage function by finding the temporal difference (TD) error between the target reward and the estimated value function $V^{\pi}(s_t)$, given as,

$$A(s_t, a_i;\theta, w) = r_t + \gamma V(s_{t+1};w) - V(s_t;w) \tag{11}$$

The advantage function represents the suitability of an action in term of the return compared to the other actions. The actor network updates itself using the equation,

$$\begin{aligned} d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi_{\theta'}(a_i \mid s_t;\theta') A(s_t, a_i;\theta', w') \\ + \beta \nabla_{\theta'} H(\pi(s_t;\theta')) \end{aligned} \tag{12}$$

where entropy parameter describes the uncertainty of the action according to a policy. For the higher value of the entropy means agent is taking more random actions. Without the entropy term the agent takes the priority actions corresponding to the high reward and converges to the locally optimum policy rather than achieving the global optimum. Whereas the parameter β is a regularization term that tries to achieve an exploitation-exploration trade-off by controlling the strength of the entropy. Furthermore, the critic model is updated using the equation,

$$dw \leftarrow dw + 2A(s_t, a_k;w, w_c) \nabla_{w'} A(s_t, a_k;\theta', w_c) \tag{13}$$

$$dw \leftarrow dw + \frac{\partial A(s_t, a_i;\theta', w')^2}{\partial w'} \tag{14}$$

Here, the critic update monitors the advantage function to minimize the error between estimated and target rewards.

4.5 System consumption problem

Tasks generated by the agents can be added in the local queue or offloaded to the nodes in the neighborhood. The decision selection to offload or enqueue locally is a complex function. Decision optimization can effectively utilize the computation resources resulting in lower residence time. The objective of the multiagent Deep reinforcement is to minimize the expected discounted future utility such that

$$\textbf{P1:}\ \text{minimize}\ V^{\pi}(s = s_t) = \sum_{i=0}^{T} E(R_t|(s_t, a_i)) \tag{15}$$

$$\begin{aligned} &\text{subject to } badhboxa_i \in \{0,1\}, \forall i \in T \\ &badhbox \sum w_j \leqslant W, j \in E \\ &\textbf{C3: } A(s_t, a_i; \theta, w) \approx 0; \forall T \text{ s.t} \pi^*(s) = a^* \end{aligned} \tag{16}$$

As discussed, Equation (14) is the expected reward for following the policy π from state s_t. It corresponds to the value estimate that in fact represents the waiting times of tasks in queue. The goal is to minimize this value estimate which in turn decreases the staying time of tasks in the queue. **C1** denotes the decision tuple $a_i = \{0, 1\}$ with local and offload actions, respectively. Agents prefer those offloading actions which help to achieve the objective function i.e. minimization of residence time. Initially, there is more exploration and agents take random action because of the ignorance of environment As the agent gets familiar, actions become more deterministic in nature moving closer to the objective. **C2** denotes that the workload w should not exceed the total available capacity W. **C3** denotes the advantage function which describes that the difference between the cumulative reward and the estimated value function should be close to zero. This is achieved when an agent follows an optimal policy π^* that leads to optimal actions a^* based on learned weights θ and w.

Notably, the vehicular environment is a limited knowledge sharing environment so it is challenging to select the optimal tasks offloading decisions that result in higher delays and slow convergence. However, as discussed in [40] federated learning serves this purpose where learning is done at multiple tiers and it avoids the overhead in knowledge sharing by only exchanging the gradients across the upper and the lower tiers. This results in faster convergence and higher system scalability. Similarly, we employed the A3C algorithm which runs multiple simulation environments in parallel. This enhanced interaction results in improved decision-making by exploring a bigger part of the state space in less time.

5 Proposed federated learning architecture for computation offloading

In this work, we propose a federated task offloading framework based on multi-agent DRL. The proposed solution lays down a multi-tier architecture, where local and global models mutually share knowledge to make well-informed task offloading decisions. As mentioned earlier, each vehicular agent captures the dynamics of the underlying vehicular environment and trains a model locally by interacting with it. Moreover, the agents continuously receive updates from an aggregated global model located at the edge device to improve its local model accordingly. As discussed, the models are learned locally and retained on the vehicular nodes. Federated learning helps to reduce the communication delays by training locally at the vehicular nodes. It also improves the privacy of the vehicles and data as a small amount of critical data is shared across the resource sharing environment and the training is done on the vehicular nodes. This preserves the privacy of the vehicles and the trained model is more robust as there are less chances of the data modifications as only

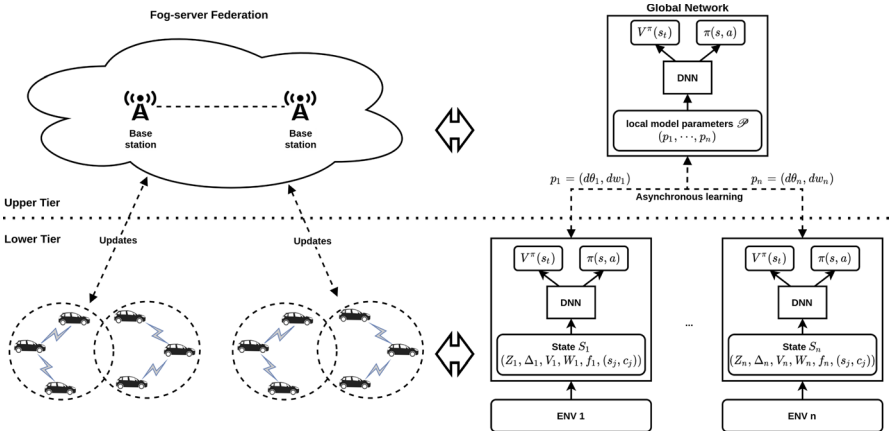

Fig. 1 Federated learning framework showing multi-tier network. The vehicles at the lower tier communicate with each other as well as the nearby connected fog node

gradients are shared across the networks. Trained model is more robust and there are less chances of the data modifications due to limited knowledge sharing.

The workflow of the proposed federated architecture is illustrated in Fig. 1, consisting of two tiers. The upper and the lower tiers. Upper tiers are based on the federation of the fog servers whereas lower tiers contain the vehicular nodes. The vehicular agents at the lower tier take the state as their respective actor and critic network learns the policy and the value functions, respectively. The updates are then shared to the fog servers at the upper tier which aggregates the actor and critic updates of all the vehicular agents in the transmission range. This mutual sharing of the updates between the lower and the upper tier improves the offloading decision model.

We consider a distributed and multi-tier learning framework for a multi-agent scenario where $\mathcal{N}$ vehicular agents jointly compete for computational resources. As discussed, we are running multiple instances of A3C algorithms and each vehicle explores the state space in an independently running A3C instance of the entire vehicular environment. Each agent tries to learn a policy π that helps it to minimize the task residence times in a highly dynamic resource-sharing environment. Unlike other DRL solutions that either use experience replay or deploy a centralized architecture, we use a distributed strategy where learning is done at multiple tiers simultaneously as shown in Fig. 1. Consequently, this leads to better decision-making in a continuous state space environment with faster rate of convergence, which is suitable for many different delay-tolerant vehicular services. As mentioned above, the learning process works at two levels elaborated as follows:

- *Lower-tier Learning* Every agent joining the vehicular network employs a local actor-critic network. The network allows the agent to learn its own separate local model. For initiation, the agent resets its local network's weights (θ_i, w_i) to ones from the global actor-critic network. Thereafter, it starts learning in an online manner by interacting with its environment. That is, upon

task arrival, the agent's actor network takes an offloading action a_i following a policy $\pi(s) \rightarrow a$, subsequently, earning a reward r_i for the action taken. The local critic network criticizes the actor's actions by estimating a state value $V^{\pi}(s_t)$. This estimated value is used by the actor network to update its offloading policy, meaning by exploring a new set of offloading actions. This collaboration between the actor and critic networks help to learn a local task offloading model. The agents follow local policy to interact with the environment for specific time steps t_{max} until a terminal state is reached. After the end of each episode, agents compute local value and policy loss, get gradients from losses, and update global network with the gradients.

- *Upper-tier Learning* At the end of every episode, the weights of the locally-learned offloading model, $(\theta'_1, \theta'_2, \cdots, \theta'_n, w'_1, w'_2, \cdots, w'_n)$, are shared with a global actor-critic network. The episode ends if a stopping criterion is reached or the game has ended. In this work, we implement the global network across a federation of fog-servers. Note that one federate is delegated the learning of the global network, possibly an underloaded fog-server with low residence time ($\arg\min_{i \in F} Z_i$). The remaining federates are responsible for cross-tier parameter exchange. The global network aggregates weights of the local model and shares an updated global model, $(\theta_1, \theta_2, \cdots, \theta_n, w_1, w_2, \cdots, w_n)$, with the agents. Due to the inherent asynchronous nature of the algorithm, the global network proceed without local updates. Hence, a diverse interaction and collaborative knowledge sharing between the two tiers helps the algorithm converge quickly, stabilizing the system performance by minimizing the expected discounted future utility $E(R_t|(s_t, a_i))$ that corresponds to the residence time of the tasks in the queue.

So, vehicles select the task offloading decisions based on the accumulated reward which relates to residence time of the tasks. As discussed, vehicles choose an action to offload tasks to the neighbors and each action is associated with the reward which the vehicle accumulates. The offloading vehicle collects the state of the vehicles at the run time and checks their accumulated reward by analyzing the residence time of the tasks. The vehicle with the lowest task residence time accumulates more reward and becomes the destination node. After the model is trained, the solution uses the learned weights and the probability distribution to select the offloading decisions at the run time for different arrival rates.

As discussed, the optimization problem is solved at the lower tier where vehicles learn the task offloading model in a distributed fashion. As we have employed the A3C algorithm which is inherently distributed in nature by running multiple simulation environments in parallel. Similarly, the vehicular agents are also interacting in their respective simulation environments. Each vehicle interacts with the environment, collects experience in an online fashion and optimizes its offloading decisions. The upper tier is responsible for the aggregation of the parameter received from the lower tier and return the updates back to the lower tier. The upper tier is helping the agents to come up with a more diversified training experience by accumulating the updates for all the vehicles in a simulation scene.

Algorithm 1 Proposed A3C Based Task Offloading Solution

procedure MAIN(self)
  if Training Model !exists **then**
    $x \leftarrow$ LowerTierLearning() ▷ Agents Online Learning
  end if
end procedure
procedure LOWERTIERLEARNING($V_1, V_2, V_3, ..., V_n$)
  Get Concurrent Threads
  $GlobalParameters \leftarrow (\theta, w)$
  $LocalParameters \leftarrow (\theta', w')$
  $d\theta = dw = 0$ ▷ Zero Gradient initialization
  $\theta' = \theta$ and $w' = w$ ▷ reset using global weights
  $x \leftarrow$ AgentExperience()
end procedure
procedure AGENTEXPERIENCE(self,s,a,r,s')
  s = sample state for an agent v_i
  for $t \leq t_{\max}$ **do**:
    while ($episode_i$!= End) **do**:
      $Actor \leftarrow \pi(s;\ a_i)$ ▷ Actor Takes Action
      $r_t(s, s')$ ▷ Gets reward and move to the next state
      Update $t = t + 1$
    end while
    $Critic \leftarrow \mathcal{V}(S_t;\ w)$; ▷ Critic Estimate
    Gradient Computation w.r.t θ' : $d\theta \leftarrow FromEquation$ **??**
    Gradient Computation w.r.t w' : $dw \leftarrow FromEquation$ 14
    $x \leftarrow$ UpperTierUpdate() ▷ Global Model Update
  end for
end procedure
procedure UPPERTIERUPDATE(dθ, dw)
  $Global_a \leftarrow$ dθ ; $Global_c \leftarrow$ dw ▷ Asynchronous Local to Global updates
end procedure

As discussed, the proposed multi-tier architecture in a multi-agent setting have several advantages. First, the architecture achieves learning diversity with agent interactions across multiple locally distributed environments. Second, the asynchronous learning helps agents to learn with faster convergence, providing broader exposure to underlying resource-sharing environment. Third, the learning at the upper tier is performed across a fog federation, which provides an extended view of the environment by aggregating knowledge shared by the lower tier. Notice that unlike the upper tier, lower tier has lower visibility due to its limited interactions with its immediate surroundings only. This interactive learning guarantees faster convergence to global minima, meaning that collaborative learning happens with actions based on global policy rather than solely on local knowledge. Vehicular network has some inherent limitations as the vehicle is connected to few vehicles and has limited perception. In this way the decisions making only occurs at one layer i.e., the lower tier which can result in sub-optimal decisions due to limited connectivity and perception of the vehicles. The upper tier consisting of fog servers has connectivity with the large number of the vehicles, so it can receive the learned data from the vehicles and other fog servers to provide an enhanced perception for better offloading decisions. This mutual learning helps the vehicles to capture the large state space in an efficient way. Fourth, the tiered architecture improves the utilization of communication channel because the models are learned locally with limited information exchange with the global network at the upper tier. Fifth and last, the

proposed work effectively utilizes communication resources by independent learning and limited knowledge exchange across the resource sharing environment. This collaborative learning framework ensures improved decision-making, which corresponds to improved task residence times in the vehicular resource-sharing network.

To conclude, the optimization problem is solved at the lower tier where vehicles learn the task offloading model in a distributed fashion. As we have employed the A3C algorithm which is inherently distributed in nature by running multiple simulation environments in parallel. Similarly, the vehicular agents are also interacting in their respective simulation environments. Each vehicle interacts with the environment, collects experience in an online fashion and optimizes its offloading decisions. So, two processes boost the performance of our algorithm. The first process is running multiple simulations environments in parallel which explore the bigger part of the state space at a particular instance whereas the second process gives the better visibility of the environment by aggregating and resharing the updates at the upper tier.

6 Performance evaluation

For experimental simulation of the proposed framework, we use SUMO traffic simulator with Manhattan road and mobility model. The simulation parameters and system specifications are mentioned in Table 3.

As discussed, our simulation scene contains 100 vehicles and 9 fog nodes. As our proposed algorithm runs multiple simulation environments in parallel so for that purpose the system should have multithreading enabled to run multiple simulations. We have used the Lenovo system Intel Core i7 with a 2.4GHz processor and 8GB RAM. The system uses two of its threads to run two parallel simulation scenes. To simulate our model, we generate the tasks with different computing requirements and nodes with diverse computing capabilities to train the DRL model. We

Table 3 Simulation parameters

Parameters	Values
Simulation time/area	1000 time steps/900×900m
Vehicle/fog clock frequency	0.6–1.0 GHz/2.0 GHz
#Vehicles/#fog/#parked	100/9/20
Vehicle/fog transmission range	100m/200m
Mobility model/vehicle speed	Manhattan/Random
Task cycles/data size	0.2–1.0 Mbits/1–6 MBs
Transmit power/white Gaussian noise	$1W/10^{-3}$[23]
V2V/V2R/V2P bandwidths	1.0/2.0/1.0 MHz [41]
NN layers/Activation Function	(32,16,8,4)/tanh/Softmax
Learning Rate/Discount Factor	$10e^{-5}$/0.955 [42]
System	Intel Core i7 2.4GHz 8 GB
OS	Windows 8.1 64-bit

implement the federated learning algorithm using TensorFlow 1.13.1 and Python 3.7. The vehicular environment is simulated at varying arrival rates to provide learning diversity to the learning algorithm such as from $60 \leq \lambda \leq 200$. Specifically, the neural network comprises a four (4) layered DNN model implementing an actor-critic network to learn a local task offloading policy. The input layer takes in the environment state as 32 neurons whereas the output layer uses four (4) neurons corresponding to four different offloading decisions using softmax activation. The two hidden layers comprise sixteen (16) and eight(8) neurons, respectively. Selection of the number of neurons in a hidden layer is a hyper parameter. We analyzed the behavior of the model with the different number of the neurons in a hidden layer. The network uses tanh activation function and Adam optimizer.

Selecting the optimal number of the hidden layers is challenging and it can decrease the learning capabilities of the algorithm. Very few numbers of hidden layers in the input layers result in underfitting and a large number of neurons in the hidden layers results in the overfitting. To overcome the phenomena of overfitting and underfitting we choose an optimal number of layers which helps us to learn the task offloading parameters in an effective way [43, 44]. There is not a specific rule to select the number of hidden layers; however, the number of hidden layers is usually in between the size of the input layers and output layers. The number of hidden layers should be quite less than the number of the input layers. As the tanh is symmetric around 0 so it results in faster convergence compared to the sigmoid function. Similarly, the tanh is steeper and has higher gradient which helps the simulation to converge faster [45].

As discussed, there are multiple agents simultaneously exploring separate parts of vehicular state space. It means that due to the inherently distributed nature of the A3C algorithm, multiple simulation environments can run concurrently in different threads which are managed by the algorithm on its own. Due to multiple simulation environments the vehicles can take different actions at the same time and observe their impact. It means we are able to explore the bigger part of the state space due to the distributed nature of the algorithm. As the vehicles are learning in an online fashion so locally learned models are aggregated at the base stations.

6.1 Online learning

As discussed, our proposed solution is based on online learning where data is generated at the run time. A3C algorithm collects recently generated experience from the environment to train a task offloading model. Tasks offloading phenomena for the vehicular system is a new paradigm so a robust dataset capturing a complete state space of the vehicular system is not available. Vehicular network is a limited information sharing environment and the topology of the network is highly dynamic. For that purpose, generating a dataset that considers all aspects of the vehicular system is a challenging task. Similarly, a vehicular system is a large or continuous state space environment which requires a large sample space to generate a robust or complete dataset. So, the proposed approach is an online learning-based solution which generates the data at the real time and learns the task offloading preference. It

is inherently different from experience replay type approaches where experience is stored in a buffer where a batch of experiences are randomly chosen from to learn the offloading decisions [46].

For comparison, we use two algorithms as follows:

1. **Stochastic Selection algorithm (SA)** – selects a destination node for task offloading stochastically among a set of computational nodes within its data-transmission range. Notably, the resources are uniformly selected from the set of fog-servers, parked vehicles and mobile vehicles [47].
2. **Greedy Algorithm (GA)** – greedily selects a nearest destination node for task offloading among the available computational resources within its data-transmission range [23].

6.2 Results and discussion

6.3 Residence time

Is the total time a task spends in queue before it gets provisioned a resource for execution. Good decisions always lead to lower residence times. Proposed approach is compared with the two other baseline approaches to analyze the residence time trends for different values of the arrival rate in Fig. 2. At the lower value of the arrival rate, all methods almost behave in the same manner. However, after the arrival rate $\lambda = 140$ the trend becomes more significant. Our approach seems to exhibit a decrease of about 58.6% at $\lambda = 200$. The reason for this decrease is the improved decision selection of the proposed approach that results in the smaller queues and the lower end-to-end delay.

6.4 End-to-end delay

Denotes the total network delay which includes the task transmission delay to the recipient node, queuing delay and the computation delays at that recipient node. The end-to-end delay of our proposed scheme is lowest as depicted in Fig. 3. The proposed approach indicates a decrease of 57.11% at $\lambda = 200$. The overall decrease in the end-to-end delay reveals the optimal decision selection in a resource sharing

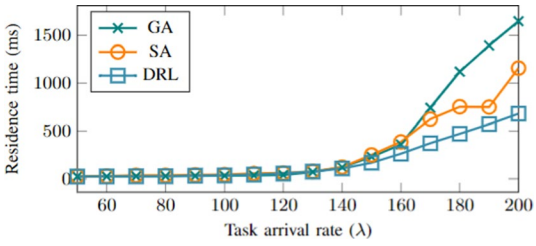

Fig. 2 Mean residence time at varying task arrival rates

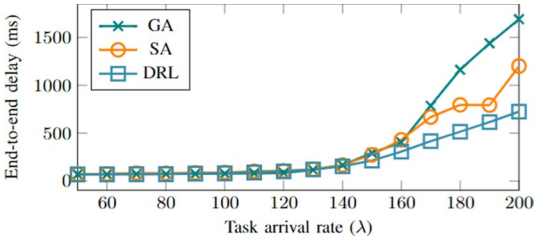

Fig. 3 Mean end-to-end delay

environment whereas, other methods take the inefficient decisions, increasing the overall delay and queue lengths.

6.5 Transmission delay

Is the delay induced due to the available data rate on the wireless communication channel. Recall, it has an impact on the residence time. Figure 4 shows the total transmission delay for the DRL approach compared against the GA and SA methods. Here, GA exhibits the lowest transmission delay as it offloads in the nearest neighborhood, however, it leads to the higher residence time. Notably, the proposed scheme shows a decrease of 8.5% compared to GA and there is an increase of 105.67% compared to the SA. DRL approach effectively utilizes the under-utilized resources of the neighboring nodes by intelligently deciding the offloading preferences.

6.6 System cost

Is defined as the sum of local execution and the offload execution. Figure 5 shows the average system cost for different offloading schemes compared to the proposed solution. Evidently, both GA and SA incur the highest cost due to their workload-in cognizant decision model. That is, for $\lambda = 200$, SA and GA show an increased cost of 16% and 56%, respectively. As discussed, the proposed scheme successfully decreases the residence time by 58.6% for $\lambda = 200$, which in turn, improves the overall system cost of the resource-sharing environment.

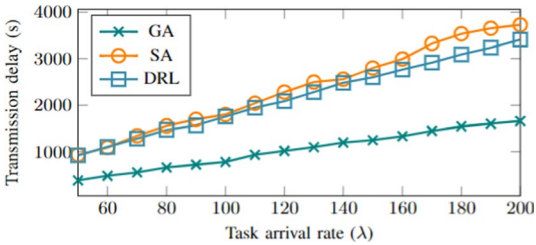

Fig. 4 Total transmission delay

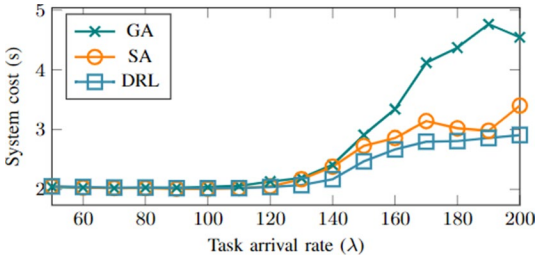

Fig. 5 Mean cost per task at varying task arrival rates

6.7 Delivery rate

Is the ratio of the total tasks offloaded to the total computed and delivered back to the offloading nodes. As discussed in Fig. 6, significant drop in the delivery is noted for the GA and the SA sachems. At $\lambda = 200$ there is a drop of 45.9% and 20.4% for GA and SA, respectively. This substantial drop in deliveries is due to the unbalanced workload distribution which causes the higher queue lengths leading to more failures. GA incurs the highest failures as more workload is diverted to the neighboring nodes, increasing the queue lengths and failures.

Figure 7 indicates the total number of tasks delivered at task arrival rate $\lambda = 200$. The proposed scheme delivers around 15K(15000) tasks whereas GA and SA deliver 17K(17000) and 14K(14000) tasks respectively. Moreover, 9K tasks are delivered through single hop that is out of the total 15K delivered for the proposed approach. Recall, delivery is the combination of the one hop and two hop deliveries. The highest number of the first hop delivery compared to SA and GA with the 8K each means that it uses the resources of the proximal nodes more efficiently. Subsequently, the proposed approach unveils the lowest number of failures with the decrease of 69% and 53% in contrast to the GA and SA. Consequently, with the

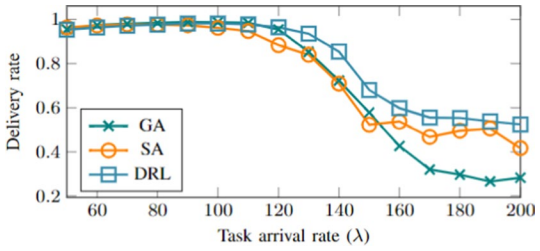

Fig. 6 Result delivery ratio with the varying arrival rate

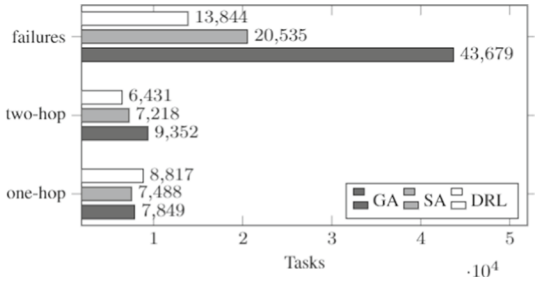

Fig. 7 Task delivery statistics for task arrival rate $\lambda = 200$

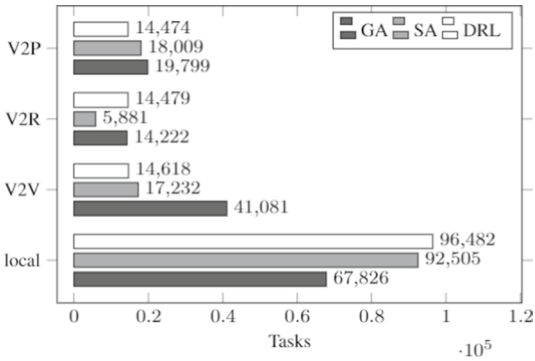

Fig. 8 Task statistics for task arrival rate $\lambda = 200$

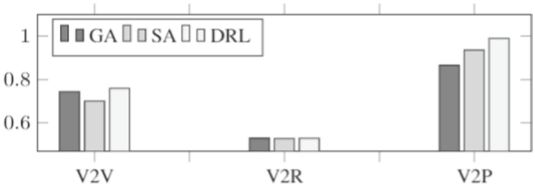

Fig. 9 Utilization ratio for task arrival rate $\sigma = 200$(task/s)

increasing arrival rates the SA and GA approach performs an unbalanced workload distribution, decreasing the delivery count and improving the failures.

6.8 Workload distribution

Figure 8 exhibits the total number of tasks allocated and executed by the system. The stats at $\lambda = 200$ show that the proposed offloading solution distributes tasks to ensure effective usage of available resources. Further breakdown shows that the proposed approach has the highest number of locally executed tasks, an increase of 30% and 4.1% compared to the GA and the SA schemes. The proposed approach uses optimal allocation of neighboring resources by allocating 15K tasks to each destination. Although the GA approach seems to be better utilizing the neighboring mobile nodes with an increase of 65% but results show a performance degradation in terms of the lowest delivery costs whereas the proposed results show improved delays and result deliveries across resource sharing framework.

6.9 Utilization ratio

Defines the ratio of the total number of tasks computed to the total number of tasks received by the available computing nodes. It can be observed that the proposed DRL based approach effectively utilizes the resource sharing environment. Figure 9 describes the ratio of three different types of the scenarios i.e. V2V, V2F, and V2P. It can be seen that the task utilization efficiency of the proposed approach is highest compared to the GA and SA. For all the three cases, the proposed approach better

utilizes the resources: for V2P(5.35% improvement) and for V2V(8.5% improvement). Recall, these are the stats for the congested state of the system

Figure 10, top and middle plots, represents actor- and critic-network loss, respectively. Initially, the actor network explores by taking random actions while observing the reward from the environment. The actor adjusts its offloading policy based on the accumulated reward, subsequently, it selects future action contributing to higher reward with lower residence times. Moreover, the critic-network loss is related to value estimation that minimizes the advantage error by reducing the difference between the target reward and value function exhibited in the bottom rewards plot. This also guides and critiques the actor network to review its policy.

7 Conclusion and future work

In this work, we presented a resource allocation framework based on a federated Multi-agent based deep reinforcement learning. The proposed model simultaneously learns the decision model at multiple tiers which results in fast convergence for a delay sensitive resource sharing environment. The simulation results show that our multi-agent deep reinforcement learning framework achieves improved task residence times, better end-to-end delay, enhanced results deliveries and improved resource utilization in a heterogeneous environment. The proposed work improve the utilization ratio; in V2P 5.35% and V2V 8.5% improvement is noted; Further, 9K tasks are delivered through single hop. Moreover, the local execution has been increased by 30% in comparison with GA.

In future, we are planning to extend this work to incorporate more sophisticated DRL techniques for comparison. The framework will be made available to the community through the GitHub repository. Further, other techniques can be adopted

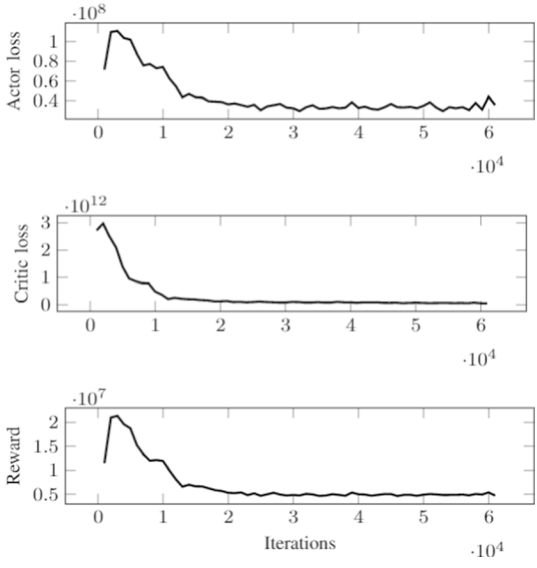

Fig. 10 The convergence trend of the proposed DRL task offloading scheme for task arrival rate $\lambda = 200$. Top: represents the total actor-network loss; Middle: represents the total critic-network loss; and Bottom: represent the accumulated reward over time

to improve sampling efficiency, transmission delays, and overall collaborative system performance. The large number of gradient updates shared across the resource sharing vehicular environment could consume some communication resources can be improvised to reduce the network utilization. Although training locally ensures the privacy of the data shared across the tiers, it is vulnerable to gradient spoofing attacks because of the updates shared in plain

Funding Information No funding was received for conducting this study.

Data Availability Data availability - There is no data associated with this work.

Declarations

Conflict of interest All authors certify that they have no affiliations with or conflict or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. Ma L, Wang X, Wang X, Wang L, Shi Y, Huang M (2021) Tcda: Truthful combinatorial double auctions for mobile edge computing in industrial internet of things. IEEE Trans Mobile Comput 2:58
2. Liu S, Guo L, Easa SM, Chen W, Yan H, Tang Y (2018) Chaotic behavior of traffic-flow evolution with two departure intervals in two-link transportation network. Discr Dyn Nat Soc 68:45
3. Raza S, Wang S, Ahmed M, Anwar MR (2019) A survey on vehicular edge computing: architecture, applications, technical issues, and future directions. Wireless Commun Mobile Comput 2019:865
4. Ranadheera S, Maghsudi S, Hossain E (2017) Mobile edge computation offloading using game theory and reinforcement learning. http://arxiv.org/abs/1711.09012
5. Ning Z, Dong P, Wang X, Rodrigues JJ, Xia F (2019) Deep reinforcement learning for vehicular edge computing: An intelligent offloading system. ACM Trans Intell Syst Technol (TIST) 10(6):1–24
6. Qi Q, Wang J, Ma Z, Sun H, Cao Y, Zhang L, Liao J (2019) Knowledge-driven service offloading decision for vehicular edge computing: a deep reinforcement learning approach. IEEE Trans Veh Technol 68(5):4192–4203
7. Zhan W, Luo C, Wang J, Wang C, Min G, Duan H, Zhu Q (2020) Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing. IEEE Internet Things J 7(6):5449–5465
8. Zhu Z, Wan S, Fan P, Letaief KB (2021) Federated multi-agent actor-critic learning for age sensitive mobile edge computing. IEEE IoT J 9:1053
9. Tian G, Ren Y, Pan C, Zhou Z, Wang X (2022) Asynchronous federated learning empowered computation offloading in collaborative vehicular networks. In: 2022 IEEE Wireless Communications and Networking Conference (WCNC), pp. 315–320 . IEEE
10. Shinde SS, Bozorgchenani A, Tarchi D, Ni Q (2021) On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems. IEEE Trans Veh Technol 71(2):2041–2057
11. Kang J, Xiong Z, Niyato D, Zou Y, Zhang Y, Guizani M (2020) Reliable federated learning for mobile networks. IEEE Wireless Commun 27(2):72–80
12. Verbraeken J, Wolting M, Katzy J, Kloppenburg J, Verbelen T, Rellermeyer JS (2020) A survey on distributed machine learning. ACM Comput Surv 53(2):1–33
13. Lu Y, Huang X, Zhang K, Maharjan S, Zhang Y (2020) Low-latency federated learning and blockchain for edge association in digital twin empowered 6g networks. IEEE Trans Industr Inform 17(7):5098–5107
14. Thrun S, Littman ML (2000) Reinforcement learning: an introduction. AI Magaz 21(1):103–103

15. Lewis FL, Vrabie D (2009) Reinforcement learning and adaptive dynamic programming for feedback control. IEEE Circuits Syst Mag 9(3):32–50
16. Van Seijen H, Fatemi M, Romoff J, Laroche R, Barnes T, Tsang J (2017) Hybrid reward architecture for reinforcement learning. http://arxiv.org/abs/1706.04208
17. Spielberg S, Gopaluni R, Loewen P (2017) Deep reinforcement learning approaches for process control. In: 2017 6th International Symposium on Advance Control of Induction Processes (AdCONIP), pp. 201–206. IEEE
18. Zhang Y, Yao J, Guan H (2017) Intelligent cloud resource management with deep reinforcement learning. IEEE Cloud Comput 4(6):60–69
19. O'Shea TJ, Clancy TC (2016) Deep reinforcement learning radio control and signal detection with kerlym, a gym rl agent. http://arxiv.org/abs/1605.09221
20. Gu S, Holly E, Lillicrap T, Levine S (2017) Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE Int Conf Robot Autom, pp. 3389–3396 . IEEE
21. Xiong X, Wang J, Zhang F, Li K (2016) Combining deep reinforcement learning and safety based control for autonomous driving. http://arxiv.org/abs/1612.00147
22. Mekrache A, Bradai A, Moulay E, Dawaliby S (2021) Deep reinforcement learning techniques for vehicular networks: recent advances and future trends towards 6g. Veh Commun 25:100398
23. Yang C, Liu Y, Chen X, Zhong W, Xie S (2019) Efficient mobility-aware task offloading for vehicular edge computing networks. IEEE Access 7:26652–26664
24. Rahman AU, Malik AW, Sati V, Chopra A, Ravana SD (2020) Context-aware opportunistic computing in vehicle-to-vehicle networks. Veh Commun 24:100236
25. Sun Y, Guo X, Song J, Zhou S, Jiang Z, Liu X, Niu Z (2019) Adaptive learning-based task offloading for vehicular edge computing systems. IEEE Trans Veh Technol 68(4):3061–3074
26. Zhang J, Guo H, Liu J, Zhang Y (2019) Task offloading in vehicular edge computing networks: a load-balancing solution. IEEE Trans Veh Technol 69(2):2092–2104
27. Hu J, Li K, Liu C, Li K (2020) Game-based task offloading of multiple mobile devices with qos in mobile edge computing systems of limited computation capacity. ACM Trans Embedded Comput Syst 19(4):1–21
28. Tang C, Xia S, Li Q, Chen W, Fang W (2021) Resource pooling in vehicular fog computing. J Cloud Comput 10(1):1–14
29. Gao M, Cui W, Gao D, Shen R, Li J, Zhou Y (2019) Deep neural network task partitioning and offloading for mobile edge computing. In: 2019 IEEE Global Communication Conference, pp. 1–6 . IEEE
30. Hu L, Tian Y, Yang J, Taleb T, Xiang L, Hao Y (2019) Ready player one: Uav-clustering-based multi-task offloading for vehicular vr/ar gaming. IEEE Netw 33(3):42–48
31. Mashhadi F, Monroy SAS, Bozorgchenani A, Tarchi D (2020) Optimal auction for delay and energy constrained task offloading in mobile edge computing. Comput Netw 183:107527
32. Chen Z, Hu J, Chen X, Hu J, Zheng X, Min G (2020) Computation offloading and task scheduling for dnn-based applications in cloud-edge computing. IEEE Access 8:115537–115547
33. Guo H, Liu J, Ren J, Zhang Y (2020) Intelligent task offloading in vehicular edge computing networks. IEEE Wireless Commun 27(4):126–132
34. Zhang Y, Zhang M, Fan C, Li F, Li B (2021) Computing resource allocation scheme of iov using deep reinforcement learning in edge computing environment. J Adv Signal Process 1:1–19
35. Shi J, Du J, Wang J, Wang J, Yuan J (2020) Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning. IEEE Trans Veh Technol 69(12):16067–16081
36. Lin B, Lin K, Lin C, Lu Y, Huang Z, Chen X (2021) Computation offloading strategy based on deep reinforcement learning for connected and autonomous vehicle in vehicular edge computing. J Cloud Comput 10(1):1–17
37. Lei L, Tan Y, Zheng K, Liu S, Zhang K, Shen X (2020) Deep reinforcement learning for autonomous internet of things: model, applications and challenges. IEEE Commun Surv Tutorials 22(3):1722–1760
38. Cui T, Hu Y, Shen B, Chen Q (2019) Task offloading based on lyapunov optimization for mec-assisted vehicular platooning networks. Sensors 19(22):4974
39. Guo H, Liu J (2018) Collaborative computation offloading for multiaccess edge computing over fiber-wireless networks. IEEE Trans Veh Technol 67(5):4514–4526
40. Prathiba SB, Raja G, Anbalagan S, Dev K, Gurumoorthy S, Sankaran AP (2021) Federated learning empowered computation offloading and resource management in 6g–v2x. IEEE Trans Netw Sci Eng 2:65

41. Raza S, Liu W, Ahmed M, Anwar MR, Mirza MA, Sun Q, Wang S (2020) An efficient task offloading scheme in vehicular edge computing. J Cloud Comput 9:1–14
42. Li M, Gao J, Zhao L, Shen X (2020) Deep reinforcement learning for collaborative edge computing in vehicular networks. IEEE Trans Cognit Commun Netw 6(4):1122–1135
43. Sheela K, Deepa SN (2013) Review on methods to fix number of hidden neurons in neural networks. Math Probl Eng. https://doi.org/10.1155/2013/425740
44. Uzair M, Jamil N (2020) Effects of hidden layers on the efficiency of neural networks. In: 2020 IEEE 23rd International Multitopic Conference (INMIC), pp. 1–6 . doi: https://doi.org/10.1109/INMIC50486.2020.9318195
45. Hayou S, Doucet A, Rousseau J (2018) On the selection of initialization and activation function for deep neural networks. arXiv
46. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
47. Alghamdi I, Anagnostopoulos C, Pezaros DP (2019) On the optimality of task offloading in mobile edge computing environments. Proc IEEE Global Commun Conf (GLOBECOM) 19:1–6

Authors and Affiliations

Balawal Shabir[1] · Anis U. Rahman[1] · Asad Waqar Malik[1] · Rajkumar Buyya[2] · Muazzam A. Khan[3]

Balawal Shabir
bilawalshabir@gmail.com

Anis U. Rahman
anis.rahman@seecs.edu.pk

Rajkumar Buyya
rbuyya@unimelb.edu.au

Muazzam A. Khan
khattakmuazzam@gmail.com

[1] School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabad, Pakistan

[2] School of Computing and Information Systems, The University of Melbourne, Melbourne, Australia

[3] Department of Computer Science, Quaid-i-Azam University, Islamabad, Pakistan