Fairness-Aware Multi-Agent Learning-based Task Offloading in Dynamic Vehicular Scenarios

Ziqi Zhou*, Agon Memedi*, Chunghan Lee†, Seyhan Ucar†, Onur Altintas†, and Falko Dressler*
* School of Electrical Engineering and Computer Science, TU Berlin, Germany
† InfoTech Labs, Toyota Motor North America R&D, CA, U.S.A.
Email: {zhou, memedi, dressler}@ccs-labs.org, {chunghan.lee1, seyhan.ucar, onur.altintas}@toyota.com

***Abstract*—Task offloading in mobile edge computing (MEC) is essential for reducing latency, balancing workload, and meeting task deadlines. In the scope of vehicular networks, the concept of a vehicular micro cloud (VMC) has been designed to handle such edge computing without the need for installed MEC infrastructure. However, the mobility vehicles and the need to fairly distribute workload make it vital to develop an adaptive and intelligent task offloading strategy. We propose a multi-agent twin delayed deep deterministic policy gradient (MATD3)-based task offloading strategy that enables vehicles to make decentralized, dynamic offloading decisions. Our solution significantly enhances overall fairness stability, and also improves delays to meet relevant deadlines. Our approach is evaluated in a single intersection scenario and a real-world traffic scenario from Nagoya. We compare our approach with a greedy and exhaustive baseline that sequentially offloads tasks to the current least-loaded vehicle. Compared to the baselines, our solution can deal with dynamic scenarios and provide long-term workload fairness in combination with reduced delays.**

***Index Terms*—virtualized edge computing, task offloading, fairness, workload balance, task migration, vehicular micro cloud**

I. Introduction

Mobile edge computing (MEC) places computing resources near users to reduce latency and local computing burdens by offloading computational tasks to nearby edge servers. In normal 5G MEC setups [1, 2], these edge nodes are stationary and can be centrally managed, allowing coordinated task offloading from users to edge servers. In order to bridge limited deployments and to provide better resource utilization, virtual edge computing (V-Edge) has been proposed, which aims at edge computing not only using 5G edge servers but also available computational resources at nearby users [3]. Thanks to advancements in cellular V2X (C-V2X) [4, 5], the concept of a vehicular micro cloud (VMC) [6, 7] has been developed, allowing vehicles within a cluster to collaboratively better process the computation tasks of each other. This decentralized system enables distributed task offloading and resource allocation without relying on fixed infrastructure.

Task offloading in edge computing but particularly in VMCs presents several challenges. We observe high vehicular mobility [8, 9] with uncertain and hard-to-predict dwell times [10]. Communication quality between vehicles and to the radio access network fluctuates significantly, and for previous solutions, the dependence on a central controller limits the applicability in many scenarios [11]. Breaking down relevant research questions, we need to understand (1) when and where tasks should be offloaded, (2) which vehicles or devices should offload tasks, and (3) what should be the migration order among tasks. Key performance indicators (KPIs) to be optimized include minimizing task completion delay, ensuring fair utilization and workload of distributed computing resources, reducing energy consumption, and maintaining robust performance under changing traffic dynamics.

In this work, we target two primary objectives: minimizing task completion delay and promoting workload fairness among vehicles, i.e., the virtual edge servers in a VMC. We propose a decentralized, learning-based task offloading strategy using multi-agent twin delayed deep deterministic policy gradient (MATD3), which enables each vehicle to make continuous offloading decisions based on its observation of the VMC environment.

We compare our method against the three baselines. First, a heuristic policy that offloads tasks to minimize task completion time, with full knowledge of the system state [12, 13]. Secondly, an exhaustive fairness-oriented method that always offloads tasks to the vehicle with the lowest current workload, similar to the work in [14, 15]. Third, we also use a random offloading approach where tasks are assigned to random vehicles regardless of context, which is often used in the related work as a baseline. To evaluate our approach under realistic conditions, we consider two types of traffic environments. A simple single-intersection scenario, which allows for a very controlled experiment, and three realistic intersections extracted from Nagoya Urban Mobility (NUMo) scenario [8]: city (downtown), a state road, and a suburban intersection. For evaluation, we use task completion time, task punctuality (on-time vs. late), Jain's fairness index over vehicular computing workloads, and utilization rates of each vehicle's computing capacity in the VMC as metrics.

Our main contributions can be summarized as follows:

- We propose a fully decentralized MATD3-based framework for task offloading;
- we particularly focus on improving the fairness in offloading decisions measured using the Jain's fairness index; and
- we present and discuss simulation results showing that our method performs robustly and adapts intelligently to changing environments, outperforming state of the art baseline approaches.

II. Related Work

Task offloading in MEC has been extensively studied [1, 16]. This includes work on energy consumption [17], minimizing latency [18], and monetary cost [19], increasing task success rates [20], and resilience [21]. Recently, machine learning (ML) techniques have been widely adopted to dynamically optimize task offloading decisions, particularly also in vehicular edge computing domain as addressed in this paper. In the following, we focus on related work on fairness-aware optimization and learning-based task offloading approaches, which are related to our work and contributions.

A. Fairness-Aware Task Offloading

Fairness in task offloading within MEC environments is crucial for preventing resource monopolization by individual users and avoiding imbalanced computational loads among edge servers. Xiao et al. [14] proposed a fairness-aware offloading strategy by formulating the task-server matching problem as a bipartite graph and solving it using the Kuhn-Munkres algorithm. Specifically, their approach defers the scheduling of less time-sensitive tasks (i.e., tasks with execution time $T_{\text{task}} < \frac{1}{2}T_{\text{max}}$) by placing them in a stack to be reconsidered in subsequent scheduling rounds. This strategy improves overall system utility, increases task completion rates, and enhances fairness compared to traditional methods.

Similarly, Zhou and Zhang [15] proposed a two-level algorithm, where the upper-level component determines a global offloading strategy, and the lower-level algorithm focuses on ensuring fairness among tasks. However, their definition of fairness is task-centric, aiming to provide equal execution opportunities for tasks, rather than considering fairness across users or servers. As such, fairness in resource distribution among different nodes is not addressed in their work.

Zhang et al. [22] propose a two-layer scheme called Two-Step Fair Task Offloading, which models the problem as a cooperative game aimed at enhancing fairness while minimizing task delays. Although their work is situated in the context of fog computing with cloud server assistance, their methodology for fairness evaluation remains relevant and insightful. Notably, they employ Jain's index to quantify fairness, focusing on energy consumption rather than workload or latency. By considering both current and historical energy consumption, their approach offers a valuable reference for future studies that incorporate energy-aware fairness in vehicular edge computing.

Building on this, Vu et al. [23] developed an energy-aware proportional fairness approach that jointly optimizes offloading and resource assignment. The proposed mechanism considers energy constraints and dynamically adapts to task demands, offering a practical balance between efficiency and fairness in resource-constrained environments.

These fairness-driven approaches lay a strong foundation for enhancing equity in edge computing. However, they often rely on deterministic or heuristic methods, which may struggle to adapt optimally in highly dynamic edge computing scenarios.

B. Reinforcement Learning-based Task Offloading

Reinforcement learning (RL) has emerged as a powerful tool to address the dynamics and complexity of decision making involved in task offloading for MEC, particularly in vehicular environments where latency, mobility, and resource constraints must be tightly managed.

Shuai et al. [24] proposed a deep reinforcement learning (DRL)-based adaptive task offloading strategy tailored for vehicular edge computing networks. Their method leverages a traditional deep Q-learning (DQL) framework – comprising a single actor and a single critic network – to make dynamic offloading decisions. The reward function is designed around a single objective: minimizing processing delay. While this approach performs adequately in delay-focused settings, it is less suitable for systems like ours that require two objectives, such as both latency and fairness. Nevertheless, their consideration of multi-hop path selection for task transmission is particularly noteworthy and aligns with one of our identified directions for future research.

To address the requirements of time-critical applications, Farimani et al. [25] proposed a deadline-aware task offloading framework based on the Rainbow DQL algorithm. Their method effectively incorporates task deadlines into the reward design – using the difference between the number of successful and failed tasks normalized by time cost. However, they assume that all tasks are offloaded to road side unit (RSU). This assumption overlooks the potential of vehicles acting as edge servers in a VMC, thereby limiting resource utilization. Nevertheless, the study's validation using real traffic data from a circular road in France [26] adds practical credibility. Their algorithm demonstrates superior performance over classical heuristics by dynamically adapting offloading decisions to meet real-time constraints, offering a strong example of deadline-aware offloading in vehicular networks.

Wang et al. [27] explored the application of the twin delayed deep deterministic policy gradient (TD3) algorithm in MEC environments. Although their study does not involve vehicular edge computing (VEC), their findings offer valuable insights. Specifically, their work demonstrates the strong learning capability of TD3 in the context of task offloading, highlighting its advantages in stability and convergence speed.

Most recently, Memedi et al. [28] presented an open source tool for RL-based studies. The running example was to optimize task-resource matching by rewarding successful task completions.

In summary, these studies collectively demonstrate that deep and actor-critic-based reinforcement learning approaches significantly enhance task offloading in vehicular and mobile edge computing environments. However, challenges remain in terms of generalization under highly dynamic topologies and coordinated learning across multiple agents. To this end, our work builds on the strengths of these RL approaches by adopting a MATD3 framework specifically designed for cooperative vehicular edge computing environments, and integrating fairness constraints directly into the learning process.

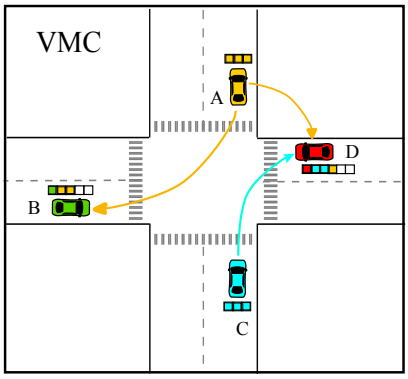


Figure 1. Task offloading in a vehicular micro cloud. In this example, vehicles B and D have spare resources, while A and C are overloaded. Therefore, A offloads their tasks to B and D and C offloads tasks to D, improving overall time efficiency and promoting a more balanced workload distribution across vehicles.

III. System Model

In the following, we introduce the system model as well as the main problem of task migration.

A. Overview of Vehicular Micro Clouds

A concrete realization of virtualized edge computing is the concept of a vehicular micro cloud (VMC) [6, 7]. A VMC comprises of vehicles in a certain region. These vehicles share available communication and computation resources for distributed edge computing. An example is depicted in Figure 1. In this case, four cars form a VMC. Vehicles can act as both task generators and mobile edge servers with varying computational capabilities, enabling efficient task processing. Tasks generated by vehicles may be heterogeneous in data size, computational complexity, and deadlines. The dynamics of all participating vehicles can be well described, e.g., in terms of their dwell times, to support task migrations schedules [10, 13].

B. Task Model

Following the notion in [13], we denote task IDs by k and use $\max\{k\}$ to represent the total number of tasks generated in the VMC. Each task varies in data size (measured in megabytes, MB), computational complexity (measured in million instructions, MI), and deadline (measured in seconds, s). The total delay T_k for task k is the sum of its communication time T_k^{c}, waiting time T_k^{w}, and processing time T_k^{p}, formulated as:

$$T_k \triangleq T_k^{\mathrm{w}} + T_k^{\mathrm{c}} + T_k^{\mathrm{p}}.$$

C. Vehicle Model

Vehicles in a VMC have heterogeneous computing capabilities (in million instructions per second, MIPS), varying workloads (in million instructions, MI), and mobility related parameters such as location and speed. We use t and i to denote time steps and individual cars, respectively. We use Veh_i to denote the i-th vehicle in a VMC and W_i represents its corresponding computing workload. For a given task, Veh^* represents the vehicle that processes it.

D. Communication Model

The transmission rate between two vehicles is calculated using the free space path loss (FSPL) model and the Shannon capacity. Let vehicle i be the transmitter and vehicle j be the receiver. At simulation time t, the transmission rate $R_{i,j}(t)$ in Mbps is computed as:

$$R_{i,j}(t) = \frac{B}{10^6} \cdot \log_2\left(1 + \mathrm{SNR}_{i,j}(t)\right) \tag{1}$$

where B is the communication bandwidth in Hz (typically 10 MHz), and $\mathrm{SNR}_{i,j}(t)$ is the signal-to-noise ratio between vehicles i and j at time t. The SNR is derived from the received signal power and noise power as:

$$\mathrm{SNR}_{i,j}(t) = 10^{\frac{P_{\mathrm{rx},i,j}(t) - N_0}{10}} \tag{2}$$

where: $P_{\mathrm{rx},i,j}(t)$ is the received power in dBm, N_0 is the noise power in dBm, P_{tx} is the transmit power, $d_{i,j}(t)$ is the Euclidean distance in meters between vehicle i and j at time t, and f is the carrier frequency.

The received power is computed using the FSPL model as:

$$P_{\mathrm{rx},i,j}(t) = P_{\mathrm{tx}} - \mathrm{FSPL}(d_{i,j}(t)) \tag{3}$$

$$\mathrm{FSPL}(d) = 20\log_{10}(d) + 20\log_{10}(f) + 20\log_{10}\left(\frac{4\pi}{c}\right) \tag{4}$$

with $c = 3 \times 10^8$ m/s being the speed of light.

E. Agent Model

In the decentralized MATD3 framework, each vehicle acts as a learning agent responsible for making task migration decisions based on observations of the environment. Unlike the centralized controller, there is no global information collection and coordination. Each agent observes the common environment while independently adapts its migration policy and updates reinforcement network parameters.

At each time slot t, an agent performs the following steps:

1) State observation: The agent observes the state s_t in the current VMC, including vehicular computing capacity and cars' workload.
2) Information Recording: Upon task generation, the agent records their task attributes (e.g., data size, deadline, computational load).
3) Action selection: The actor network $\pi_{\theta_i}(s_t)$ determines where to offload the task based on the current state s_t.
4) Learning: During training, the agent stores observations and actions and updates its network parameters.

F. Baseline Algorithms

1) Workload Balance-oriented Greedy Task Migration Heuristic: In the first baseline approach (Algorithm 1), each newly generated task is offloaded to the vehicle with the lowest current workload. Although there is no widely adopted heuristic that simply "always offloads tasks to the least-loaded server" or uses "workload fairness" as the sole objective, many existing studies take into account factors such as server idleness and current workload, such as [14, 15]. The tasks generated in the

Algorithm 1 Heuristic Offloading: Least-Workload First

Input: Set of vehicles Veh, tasks $\mathcal{T}$, current workloads W_i for each vehicle Veh_i

Output: Task offloading based on minimum workload

for each task Tsk_k in $\mathcal{T}$ **do**
  Identify generator vehicle Veh_i for task Tsk_k
  Find target vehicle with minimum workload:

$$Veh^* = \arg \min_{Veh_j \in Veh} W_j$$

  if $Veh^* = Veh_i$ **then**
    Process task locally
  else
    Offload task Tsk_k to Veh^*
  end if
  Update workload $W_{Veh^*} += MI(Tsk_k)$ and $W_{Veh_i} -= MI(Tsk_k)$
end for

Algorithm 2 Heuristic Offloading: Shortest-Completion-Time First

Input: Set of vehicles Veh, sorted tasks $\mathcal{T}_{sorted}$, current workloads W_i, CPU capacities C_i for each vehicle Veh_i

Output: Task offloading based on estimated completion time

$\mathcal{T}_{sorted} \leftarrow$ Newly generated tasks sorted by deadline
for each task Tsk_k in $\mathcal{T}_{sorted}$ **do**
  Identify generator vehicle Veh_i for task Tsk_k
  Calculate the completion time on all vehicles Veh_j:

$$T_k \triangleq T^{\mathrm{w}}_{(j,k)} + T^{\mathrm{c}}_{(j,k)} + T^{\mathrm{p}}_{(j,k)}$$

  Find target vehicle with minimum estimated completion time:

$$Veh^* = \arg \min_{Veh_j \in Veh} T^{(j,k)}_{comp}$$

  if $Veh^* = Veh_i$ **then**
    Process task locally
  else
    Offload task Tsk_k to Veh^*
  end if
  Update workload $W_{Veh^*} += MI(Tsk_k)$ and $W_{Veh_i} -= MI(Tsk_k)$
end for

current time slot are ordered by their generation time, with no pre-scheduling applied. The central controller selects the car with the minimum workload for each task. If the source vehicle already has the least workload, the task is processed locally.

2) Shortest Time-oriented Greedy Task Migration Heuristic: In the second baseline approach (Algorithm 2) [13], we assume perfect knowledge about the mobility of cars and assign newly generated tasks according to earliest deadline first (EDF). That is, we offloaded to the vehicle that will complete it in the shortest time. The task completion time is calculated following [13] and the vehicle with the minimum estimated completion time is selected for execution. This strategy aims to minimize individual task delay instead of balancing workload across the VMC.

IV. MATD3-BASED FAIRNESS-AWARE TASK MIGRATION

This section introduces our proposed MATD3 approach for decentralized task migration. Our goal is to optimize task completion time and balance workload among vehicles in a dynamic traffic environment.

MATD3 adopts a multi-agent framework where each vehicle acts as an independent agent, making task offloading decisions. Once an action is taken (e.g., task offloading), the global state, including updated workloads and task assignments, is shared across all agents within the VMC. Importantly, the twin critic networks evaluate action-state pairs across the entire environment, not just for their own agents. This global evaluation stimulates more effective learning and accelerates convergence to better offloading strategies. The decentralized nature of MATD3, with each vehicle as a decision-maker, enhances system scalability and flexibility. This design is generally well-suited to dynamic traffic conditions, where centralized control may become inconvenient due to high mobility and communication delays.

A. MATD3 Structure

Figure 2 illustrates the architecture of the proposed MATD3-based task offloading framework adapted to task offloading decision-making in VMC. In this multi-agent system, each vehicle acts as an agent, interacting with the environment, i.e., observing the VMC and obtaining the current state. An actor network within each agent generates offloading actions, while two independently parameterized critic networks use the global state and action of all agents as input, to reduce overestimation bias. However, critics do not communicate with each other's network parameters. To stabilize training, target networks – soft copies of the actor and critics – are used for stabilizing target value computation. This architecture enables decentralized decision-making across vehicles; but critics still evaluate the global situation for better and meaningful learning.

B. Reward Function

The average completion time of N tasks at time t is computed as:

$$\bar{T}_{comp}(t) = \frac{1}{N} \sum_{k=1}^{k=N} T^{\mathrm{sum}}(k, t, p(k)) \tag{5}$$

where $p(k)$ is the processing car of task k. The computational workload of a vehicle is formulated as how many of instructions it still needs to execute:

$$W(i,t) = \sum_{k=1}^{N} \mathsf{MI}(k) + (1 - \alpha_{k_0,t}) \mathsf{MI}(k_0) \tag{6}$$

where k represents the k-th task in the waiting queue of car i, k_0 implies the current being processed task, $\alpha_{k_0,t}$ implies the proportion of instructions that have been finished of $\mathsf{Tsk}(k_0)$.

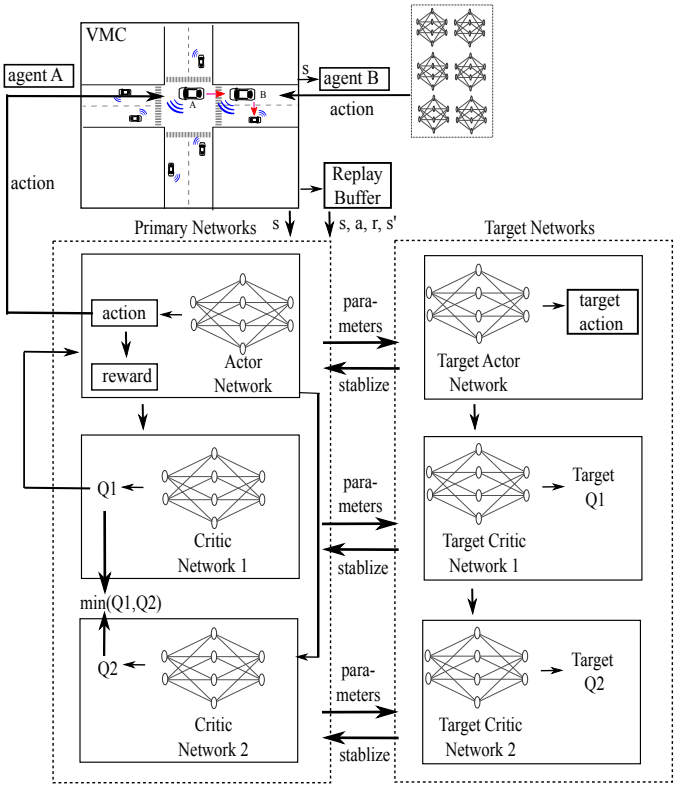


Figure 2. MATD3 Architecture of the proposed MATD3-based task offloading framework in vehicular edge computing. 1) Agents: Each vehicle (agent) interacts with the VMC, observing local states. 2) Actor network: The *actor network* generates continuous offloading actions. 3) Twin critic networks: Two independently parameterized *critics* evaluate actions to reduce overestimation bias, which a single critic cannot effectively achieve. 4) Replay buffer: The *replay buffer* stores past experiences, which are sampled to update the networks using off-policy learning. 5) Three target networks: To stabilize training, *target networks* (soft copies of the actor and two critics) are used for smoothed target value computation. The multi-agent architecture allows vehicles to learn optimal migration policies, adapting to mobility and resource variations.

To quantify how the computational workload is distributed across vehicles, we adopt Jain's fairness index [29], a widely used metric alsd in the field of task offloading [14, 22]. A higher value indicates a better-balanced task computing workload among cars, showing less likely resource overloading and implying fairness. Jain's fairness index of cars' computing workload at time t is computed as:

$$J(t) = \frac{\left(\sum_{i=1}^{i=n} W(i,t)\right)^2}{n \sum_{i=1}^{i=n} W(i,t)^2} \tag{7}$$

where $W(i,t)$ is the workload of car i at time t, and n is the number of vehicles. The reward function in the MATD3-based task migration strategy is designed to optimize both task completion time and workload fairness among vehicles. It is defined as:

$$R(t) = \lambda \left(\frac{1}{\bar{T}_{comp}(t)}\right) + (1-\lambda) J(t) \tag{8}$$

where λ is a weight parameter that balances between task completion time and fairness, $\bar{T}_{comp}(t)$ is the average task completion time across all tasks that have been completed no later than time t, and $J(t)$ is the contemporary fairness metric of the computing workload of cars at time t. The average reward value over a duration δ is

$$R(\delta) = \sum_{t=0}^{t=\delta} R(t). \tag{9}$$

The reward function encourages lower task completion time while maintaining fairness in workload distribution.

Algorithm 3 Multi-Agent TD3 Training

Input: Vehicles Veh, tasks $\mathcal{T}$, actor networks π_θ, critic networks Q_{ϕ_1}, Q_{ϕ_2}, replay buffer $\mathcal{D}$, state s, target actor networks $\hat{\pi}_\theta$, target critic networks $\hat{Q}_{\phi_1}, \hat{Q}_{\phi_2}$

Output: Trained policies for distributed offloading

Initialize each generator $Veh_i \in Veh$ with:
- Actor network π_{θ_i} (offloading decision)
- Twin critics $Q_{\phi_{i1}}, Q_{\phi_{i2}}$ (action evaluation)
- The replay buffer $\mathcal{D}$

for each timestep t **do**
  for each generator Veh_i of $\mathcal{T}$ **do**
    Each car observes $s_{t,j} \forall Veh_j$
    Select action $a_{t,j} = \pi_{\theta_j}(s_{t,j}) + \epsilon$ (exploration noise)
    Execute $a_{t,j}$, observe reward $r_{t,j}$, new state $s_{t+1,j}$
    Store $(s_{t,j}, a_{t,j}, r_{t,j}, s_{t+1,j})$ in $\mathcal{D}$
  end for
  if buffers are sufficiently full **then**
    for each vehicle Veh_i **do**
      Sample mini-batch (s, a, r, s') from $\mathcal{D}$
      Compute target

$$\hat{Q}_{\text{target}} = r + \gamma \min(Q_{\phi'_{i1}}(s', a'), Q_{\phi'_{i2}}(s', a'))$$

      Update critics: minimize loss
      if update step **then**
        Update actor π_{θ_i} via gradient ascent
        Update target networks
      end if
    end for
  end if
end for

C. Algorithm

Our proposed Multi-Agent TD3 framework operates in two distinct phases: training (Algorithm 3[1]) and execution (Algorithm 4). During the training phase, vehicles iteratively refine their offloading policies using actor-critic networks, leveraging a replay buffer to store past experiences and improve decision making through policy updates. Each vehicle selects actions with exploration noise, updates its critic networks by minimizing loss, and periodically refines the actor network.

[1]In Algorithm 3, r is the immediate reward received after taking action a in state s, γ is the discount factor that balances immediate and future rewards, s', a' is the next state and the next action; $Q_{\phi'_{i1}}$, $Q_{\phi'_{i2}}$ is the two target critic Q values, and $\hat{Q}_{\text{target}}$ is the ultimate target Q-value used for updating critic networks.

Algorithm 4 Multi-Agent TD3 Task Offloading Execution

Input: Trained actor network π_{θ_i} for each vehicle
Output: Dynamic task migration decisions following EDF

for each timestep t **do**
  $\mathcal{T} \leftarrow$ All newly generated tasks
  for each task Tsk_k in $\mathcal{T}$ **do**
    Identify generator car Veh_i for task Tsk_k
    Observe state $s_{t,j} =$ (workload W_j, MIPS)$\forall Veh_j$
    Select offloading action $a_{t,j} = \pi_{\theta_j}(s_{t,j})$
    if $a_{t,j}$ indicates offloading to a car Veh_j **then**
      Offload task to Veh_j
    else
      Process task locally
    end if
    Update vehicle workloads after execution
  end for
end for

Once training is complete, execution uses the trained policies to perform task offloading. Tasks are sorted based on deadlines and sequentially offloaded, guided by the learned policies. By dynamically adapting to network conditions and vehicle mobility patterns, the model improves both system efficiency and fairness.

V. Evaluation

Without loss of generality, we evaluate our approach in two traffic environments (cf. Figure 3): (1) a simple single-intersection scenario with new vehicles regularly driving into the VMC at a fixed frequency and (2) the Nagoya Urban Mobility (NUMo) scenario [8] as one of the most comprehensive urban traffic scenarios, providing a highly realistic vehicle mobility of the entire city of Nagoya, Japan. In particular, we selected a city, a state road, and a suburban location. These scenarios differ in terms of average vehicle count at each time step and driving speed distribution. Table I presents key traffic parameters among the selected four scenarios.

To evaluate our system under various traffic and computing conditions, we configure a set of scenarios based on both task and vehicular computational ability parameters. Table II summarizes the characteristics of tasks and computer capabilities of cars, including data size, computational complexity, processing power, and deadlines. We simulated 200 seconds with a time step of 1 s in all traffic scenarios.

Table I
Road traffic parameters across 4 scenarios

Scenario	**Avg. # of cars**	**Avg. speed** [km/h]
Single Intersection	6.00	27.86
Nagoya		
Downtown	17.20	24.864
State	10.01	22.976
Suburb	6.30	19.090

Table II
Task and car properties

Data size [MB]	**Complexity [MI]**	**CPU [MIPS]**	**Deadline [s]**
$\mathcal{U}(0.1, 1)$	$\mathcal{U}(1000, 5000)$	$\mathcal{U}(1000, 5000)$	$\mathcal{U}(1, 1.5)$

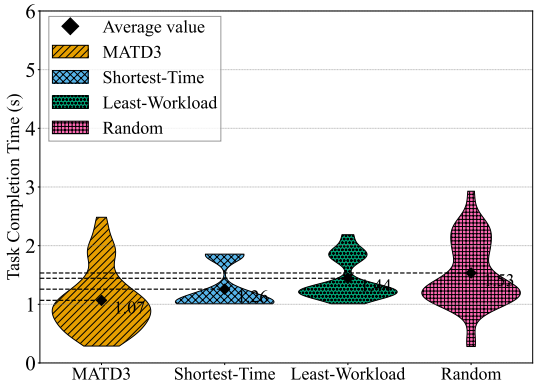
(a) Simple intersection (b) Nagoya city (c) Nagoya state road (d) Nagoya suburban

Figure 3. Selected mobility scenarios

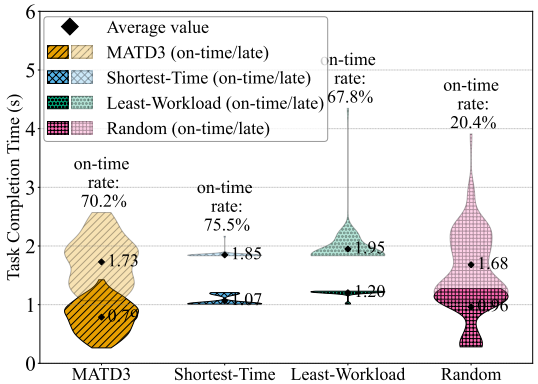
A. Simple Intersection

We first use the single intersection scenario (cf. Figure 3a). This scenario contains a single traffic light, four entry points, and 16 possible routes, including straight paths, right turns, left turns, and U-turns. The map covers an area of approximately $400\,\mathrm{m} \times 400\,\mathrm{m}$. After generating mobility for low traffic densities, the simulation outputs floating car data, including position, speed, lane, and other information about every vehicle at each time step in the simulation.

Figure 4a presents the overall distribution of task completion times, while Figure 4b further distinguishes between on-time and late task completions (both figures use violin plots to

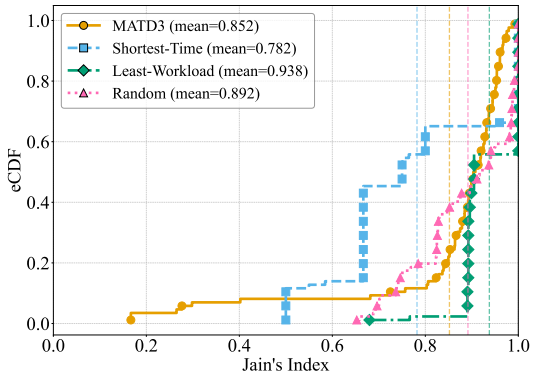


(a) Task completion time

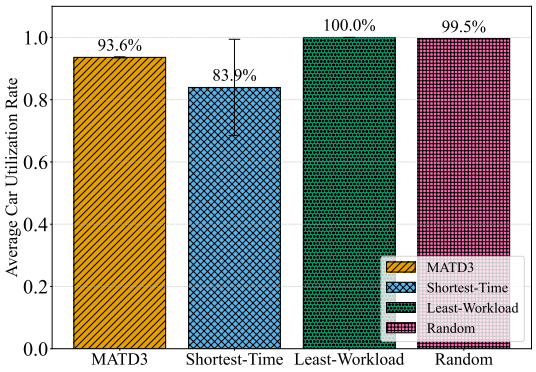


(b) Distribution of on-time / late tasks

Figure 4. Single intersection: time efficiency performance



(a) Jain's fairness index



(b) Utilization

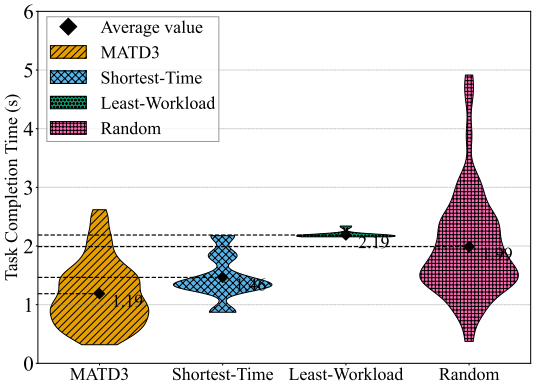
Figure 5. Single intersection: fairness and resource utilization

indicate the underlying distributions). In these figures, fully opaque colors represent tasks completed within their deadlines, whereas semi-transparent areas indicate tasks that missed their deadlines. Notably, the performance of the shortest-time approach aligns closely with the EDF-based method proposed in our previous work [13]. Our proposed method achieves a lower average task completion time, while the shortest-time approach gains a higher on-time completion rate. Both methods outperform the other two baselines in terms of time efficiency. Interestingly, the least-workload strategy also demonstrates a relatively strong on-time rate in this scenario, suggesting that, under suitable conditions, it has potential and warrants further investigation.

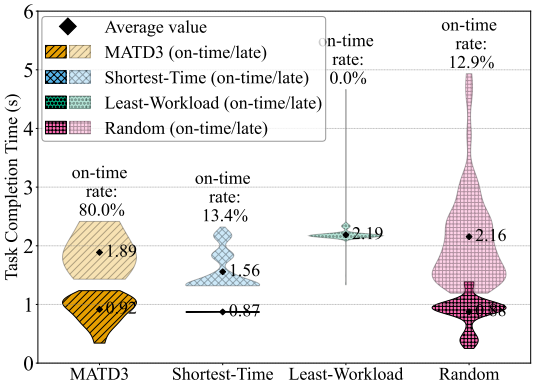
Although the Jain's fairness index (Figure 5a) for MATD3 does not surpass those of the least-workload and random approaches in the single-intersection scenario, it still achieves a satisfactory level of fairness, which, importantly, is much better than the shortest-time scheme. The computing resource utilization results (Figure 5b) reveal an interesting phenomenon: The workload balance-oriented methods consistently maximize the use of all available computing resources, resulting in a utilization rate of 1. In contrast, our proposed MATD3 also achieves high-level resource utilization.

B. Nagoya Scenarios

We focus on the Nagoya city scenario and analyze data from the morning rush hour. The simulation results for task completion time versus task deadlines, vehicle workload

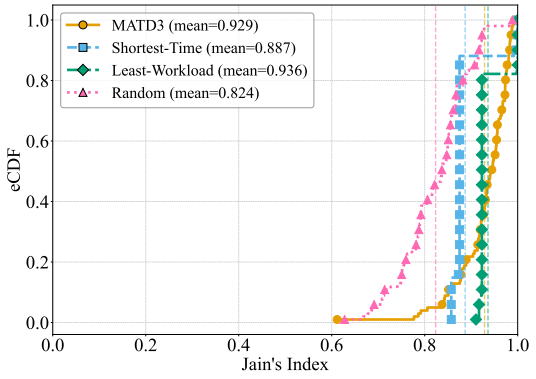


(a) Task completion time

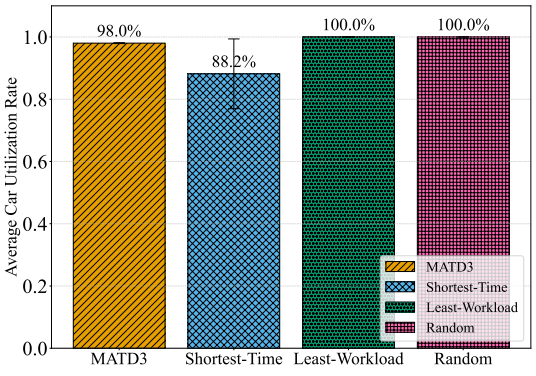


(b) Distribution of on-time / late tasks

Figure 6. Nagoya city: time efficiency performance

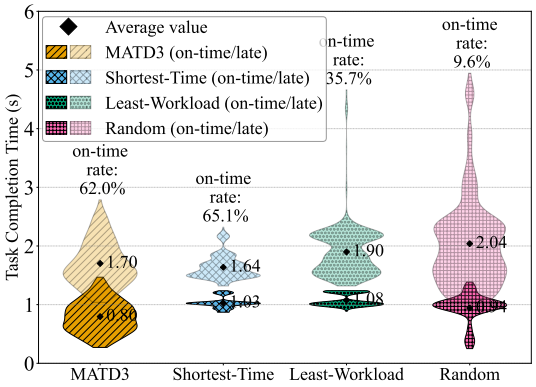


(a) Jain's fairness index

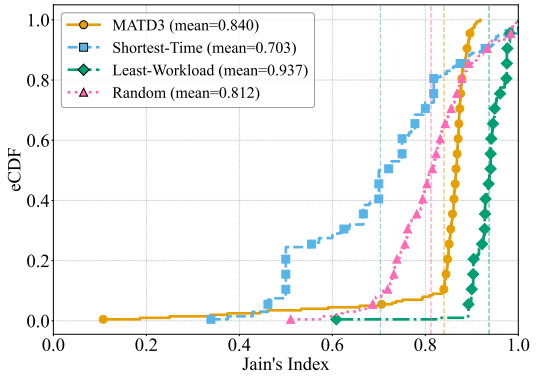


(b) Utilization

Figure 7. Nagoya city: fairness and resource utilization

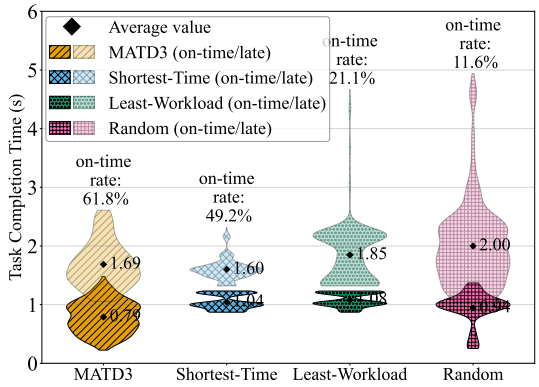


(a) Distribution of on-time / late tasks

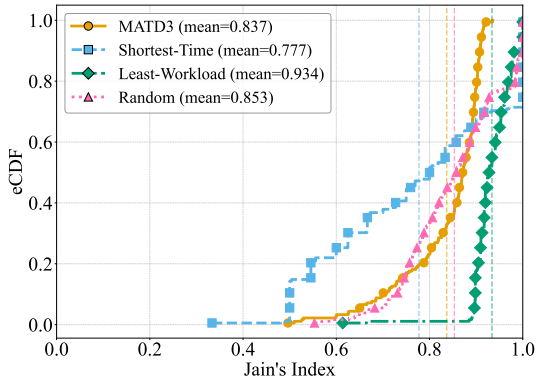


(b) Jain's fairness index

Figure 8. Nagoya state road



(a) Distribution of on-time / late tasks



(b) Jain's fairness index

Figure 9. Nagoya suburban

fairness, and its utilization rate are presented in Figures 6 and 7. As can be seen in Figure 6a, our proposed method significantly decreases task completion times compared to the other approaches. Figure 6b further illustrates that our approach effectively avoids excessively long task completion cases and achieves a an 80 % on-time completion rate. While the least-workload method achieves on average a slightly better fairness, our proposed approach still maintains a reasonable level of fairness, as shown in Figure 7a. In contrast, shortest-time and random results in noticeably reduced fairness. Similar to the single intersection scenario, the computing resource utilization rate shows that the least-workload mechanism still maintains a consistent 100% utilization rate (cf. Figure 7b). In contrast, our proposed solution achieves almost full utilization, which can be attributed to its double-objective design that considers both efficiency and fairness.

Since the other two scenarios exhibit similar performance across the four approaches, we decided to only briefly report on these results. The simulation results for task completion time versus task deadlines and workload fairness are presented in Figure 8 for the state road and in Figure 9 for the suburban intersection. As can be seen, the same effects can be seen, i.e., our MATD3-based algorithm enables lowest task completion times with best on-time ratio, while achieving second to best fairness distribution (only least-workload performs better but is much worse in task completion time).

VI. Conclusion

We presented a MATD3 framework for efficient and fair task offloading in vehicular micro clouds. By integrating actor-critic reinforcement learning with decentralized decision-making, our approach effectively addresses the trade-off between task completion delay and users' computing workload fairness. Each agent independently makes optimal offloading strategies, and its critic networks learns from global information and feedback to its actor, while target networks stabilize performance. The decision-making process dynamically evaluates environments and computes migration costs and benefits to determine the optimal offloading strategy. This approach considers both task completion delays and workload fairness among vehicles in the same VMC, which is also shown in the reward function. Overall, the proposed MATD3 effectively balances the primary objective of efficient task completion with the secondary goal of maintaining moderate workload fairness. Experimental results demonstrate that the MATD3 framework significantly improves the comprehensive performance of task migration, compared to the random, fairness-oriented exhaustive, and the shortest-time-oriented approach. The combination of multiple agents (decentralized task offloading decisions), one actor and twin critic networks with extra target networks for more stable policy updates, ensures that delay-sensitive tasks are effectively managed in dynamic vehicular environments with vehicles' workload fairness aware.

Future work may explore several directions to further improve the effectiveness and adaptability of the proposed MATD3-based task offloading strategy. These include refining the reward function to better capture fairness in workload distribution, validating the model across a wider range of real-world traffic scenarios, and enabling task partitioning or replication to increase flexibility and reliability. Additionally, embedding dwell time prediction from [13] can make the system more realistic. Multi-hop task migration, where tasks are relayed through intermediate vehicles, could also be investigated to extend offloading range and optimize latency in sparse or highly dynamic networks. By advancing these areas, we aim to enhance the robustness, scalability, and fairness of intelligent task offloading in vehicular edge computing systems.

REFERENCES

[1] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.

[2] M. Emara, M. C. Filippou, and D. Sabella, "MEC-Assisted End-to-End Latency Evaluations for C-V2X Communications," in *European Conference on Networks and Communications (EuCNC 2018)*, Ljubljana, Slovenia: IEEE, Jun. 2018.

[3] F. Dressler, C. F. Chiasserini, F. H. P. Fitzek, H. Karl, R. Lo Cigno, A. Capone, C. E. Casetti, F. Malandrino, V. Mancuso, F. Klingler, and G. A. Rizzo, "V-Edge: Virtual Edge Computing as an Enabler for Novel Microservices and Cooperative Computing," *IEEE Network*, vol. 36, no. 3, pp. 24–31, May 2022.

[4] S. A. Abdel Hakeem, A. A. Hady, and H. Kim, "5G-V2X: standardization, architecture, use cases, network-slicing, and edge-computing," *ACM/Springer Wireless Networks*, pp. 6015–6041, Jul. 2020.

[5] R. Molina-Masegosa, J. Gozalvez, and M. Sepulcre, "Configuration of the C-V2X Mode 4 Sidelink PC5 Interface for Vehicular Communication," in *14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN 2018)*, Shenyang, China, Dec. 2018.

[6] T. Higuchi, J. Joy, F. Dressler, M. Gerla, and O. Altintas, "On the Feasibility of Vehicular Micro Clouds," in *9th IEEE Vehicular Networking Conference (VNC 2017)*, Turin, Italy: IEEE, Nov. 2017, pp. 179–182.

[7] F. Dressler, G. S. Pannu, F. Hagenauer, M. Gerla, T. Higuchi, and O. Altintas, "Virtual Edge Computing Using Vehicular Micro Clouds," in *IEEE International Conference on Computing, Networking and Communications (ICNC 2019)*, Honolulu, HI: IEEE, Feb. 2019.

[8] T. Higuchi, L. Zhong, and R. Onishi, "NUMo: Nagoya Urban Mobility Scenario for City-Scale V2X Simulations," in *15th IEEE Vehicular Networking Conference (VNC 2024)*, Kobe, Japan: IEEE, May 2024, pp. 17–24.

[9] L. Codeca, R. Frank, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario: 24 Hours of Mobility for Vehicular Networking Research," in *7th IEEE Vehicular Networking Conference (VNC 2015)*, Kyoto, Japan: IEEE, Dec. 2015.

[10] G. S. Pannu, S. Ucar, T. Higuchi, O. Altintas, and F. Dressler, "Dwell Time Estimation at Intersections for Improved Vehicular Micro Cloud Operations," *Elsevier Ad Hoc Networks*, vol. 122, p. 102 606, Nov. 2021.

[11] C. Sommer and F. Dressler, *Vehicular Networking*. Cambridge University Press, 2014.

[12] Z. Zhou, Y. Tao, A. Memedi, C. Lee, S. Ucar, O. Altintas, and F. Dressler, "Optimizing Task Migration Decisions in Vehicular Edge Computing Environments," in *1st IEEE International Conference on Meta Computing (ICMC 2024)*, Qingdao, China: IEEE, Jun. 2024.

[13] Z. Zhou, A. Memedi, C. Lee, S. Ucar, O. Altintas, and F. Dressler, "Task Migration with Deadlines using Machine Learning-based Dwell Time Prediction in Vehicular Micro Clouds," *Elsevier High-Confidence Computing*, vol. 5, no. 2, p. 100 314, Jun. 2025.

[14] K. Xiao, Z. Gao, C. Yao, Q. Wang, Z. Mo, and Y. Yang, "Task Offloading and Resources Allocation based on Fairness in Edge Computing," in *IEEE Wireless Communications and Networking Conference (WCNC 2019)*, Marrakesh, Morocco: IEEE, Apr. 2019, pp. 1–6.

[15] J. Zhou and X. Zhang, "Fairness-Aware Task Offloading and Resource Allocation in Cooperative Mobile-Edge Computing," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3812–3824, Mar. 2021.

[16] K. Li, Y. Cui, W. Li, T. Lv, X. Yuan, S. Li, W. Ni, M. Simsek, and F. Dressler, "When Internet of Things meets Metaverse: Convergence of Physical and Cyber Worlds," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4148–4173, Mar. 2023.

[17] T. X. Tran and D. Pompili, "Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[18] M. D. Hossain, L. N. T. Huynh, T. Sultana, T. D. Nguyen, J. Ho Park, C. S. Hong, and E.-N. Huh, "Collaborative Task Offloading for Overloaded Mobile Edge Computing in Small-Cell Networks," in *34th International Conference on Information Networking (ICOIN 2020)*, Barcelona, Spain: IEEE, Jan. 2020, pp. 717–722.

[19] S. Cheng, T. Ren, H. Zhang, J. Huang, and J. Liu, "A Stackelberg-Game-Based Framework for Edge Pricing and Resource Allocation in Mobile Edge Computing," *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 20 514–20 530, Jun. 2024.

[20] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[21] D. Ergenç, A. Memedi, M. Fischer, and F. Dressler, "Resilience in Edge Computing: Challenges and Concepts," *Foundations and Trends in Networking*, vol. 14, no. 4, pp. 254–340, May 2025.

[22] G. Zhang, F. Shen, Y. Yang, H. Qian, and W. Yao, "Fair Task Offloading among Fog Nodes in Fog Computing Networks," in *IEEE International Conference on Communications (ICC 2018)*, Kansas City, MO: IEEE, May 2018, pp. 1–6.

[23] T. T. Vu, D. T. Hoang, K. T. Phan, D. N. Nguyen, and E. Dutkiewicz, "Energy-based Proportional Fairness for Task Offloading and Resource Allocation in Edge Computing," in *IEEE International Conference on Communications (ICC 2022)*, Seoul, South Korea: IEEE, May 2022, pp. 1912–1917.

[24] R. Shuai, L. Wang, S. Guo, and H. Zhang, "Adaptive Task Offloading in Vehicular Edge Computing Networks Based on Deep Reinforcement Learning," in *IEEE/CIC International Conference on Communications in China (ICCC 2021)*, Xiamen, China: IEEE, Jul. 2021, pp. 260–265.

[25] M. K. Farimani, S. Karimian-Aliabadi, R. Entezari-Maleki, B. Egger, and L. Sousa, "Deadline-aware task offloading in vehicular networks using deep reinforcement learning," *Expert Systems with Applications*, vol. 249, p. 123 622, 2024.

[26] M.-a. Lebre, F. L. Mouel, and E. Menard, "On the importance of real data for microscopic urban vehicular mobility trace," in *14th International Conference on Intelligent Transportation Systems Telecommunications (ITST 2015)*, Copenhagen, Denmark: IEEE, Dec. 2015, pp. 22–26.

[27] S. Wang, B. Yu, N. Wang, and W. Wang, "Research on TD3-Based Offloading Strategies for Complex Tasks in MEC Systems," in *IEEE/CIC International Conference on Communications in China (ICCC 2024)*, Hangzhou, China: IEEE, Aug. 2024, pp. 194–201.

[28] A. Memedi, C. Lee, S. Ucar, O. Altintas, and F. Dressler, "Simulator for Reinforcement Learning-based Resource Management in Vehicular Edge Computing," in *16th IEEE Vehicular Networking Conference (VNC 2025), Poster Session*, Porto, Portugal: IEEE, Jun. 2025.

[29] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation In Shared Computer Systems," Digital Equipment Corporation, Hudson, MA, DEC Research Report DEC-TR-301, Sep. 1984.