ISSN 2959-6157

Research on UAV-assisted Vehicle networking task unloading strategy based on multi-agent reinforcement learning

Fanjin Zeng

Central South University, Changsha, Hunan 410083, China
Email: fanjin1690@gmail.com

Abstract:

With the development of technology, in order to improve the user's driving experience and driving safety, there are more and more vehicle tasks with high delay requirements. Therefore, lots of researchers have paid attention to task offloading scheduling.However, as vehicle tasks become increasingly complex, a single task may consist of multiple subtasks with dependencies between them.The complex data dependencies within them make it more and more difficult to design appropriate task offloading strategies. Considering that this problem is closely related to the scenarios and requirements in the real world, this study focuses on the design of task offloading decisions in the scenario of UAV-assisted vehicle network, in which MEC servers are installed in the macro base station and UAV to provide computing resources for vehicles. We designed a task offloading strategy based on MATD3 algorithm to deal with this problem. Following simulation trials, it is evident that our approach offers notable benefits in terms of both delay and energy usage.

Keywords: Vehicle network, Multi-access edge computing, UAV, MATD3, Dependent load tasks

1. Introduction

With the yearning for a higher quality of life, the Internet of vehicles has attracted more and more attention.It plays a crucial role in building an intelligent transportation service system, improving the driving experience of users, and making driving safer. However, due to the limited computing resources of the vehicle itself, it is difficult to meet the computing requirements and low delay requirements of various computing-intensive vehicle tasks. And due to the distance limitation, there is often a huge communication delay between the vehicle and the cloud server. The application of MEC technology in the Internet of vehicles effectively solves this problem. By installing MEC servers on the macro base station, computing resources are provided for vehicles.However, The number of computing resources is usually pre-set in each macro base station. When some social events lead to a surge in traffic, it is difficult to meet the low delay requirements of all vehicle tasks only by the macro base station. In other cases, when the vehicle is far from the base station, the base station can also not meet the low delay requirements of the task.One way to solve the above problems is to use UAVs to assist the macro base station in task processing. The problems can be effectively solved by installing MEC servers on UAVs and scheduling them to a designated location to assist macro base station in completing computing tasks.

Currently, extensive research has been conducted by predecessors on the task offloading strategy in IoV scenarios. Zhang et al. [1]designed an iterative algorithm to reduce the delay of task processing as much as possible. For the vehicle edge computing and network of autonomous driving, Zhao et al.[2]design a multi-hop task offloading scheme on the premise of considering the mobility of the vehicle,which fully considers the mobility of the vehicle and timely offloads tasks to meet the real-time computing requirements of autonomous driving.Yang et al[3] mainly focus on the high speed of vehicles and the frequent connection/disconnection between vehicles. They proposed a new, efficient, and mobility-aware task offloading scheme, taking into account the mobility of vehicles. Xue et al. [3] proposed a joint computing offloading and content caching strategy for the vehicle edge computing scenario of unmanned vehicles. The study showed that this strategy can effectively improve computing performance and save energy.[4] considered the use of cloud-side collaboration to handle vehicle-mounted tasks with low delay requirements. At the same time, MEC servers and cloud servers were used to provide computing resources for vehicle-mounted tasks. Li et al. [6] proposed a multi-stage distributed task offloading strategy. Their method first se-

lects the offloading vehicle through the candidate vehicle selection mechanism and then performs task offloading, thus effectively improving the efficiency of task offloading. Zeng et al. [7] used particle swarm optimization to design task offloading strategy in IoV.

In this study, the DAG task scheduling, low delay, low energy consumption and other issues in the UAV-assisted vehicles network are expressed as an optimization problem. The problem is NP-hard and cannot be solved in polynomial time. However, the traditional algorithms are difficult to achieve the optimal solution and lack generalization. Therefore, this study intends to design a distributed cooperation scheme based on multi-agent RL method to solve this problem.

2. System Model

In this section, we first introduce the scenario architecture of UAV-assisted vehicle network, and then introduce the computing resource model and energy consumption model of MEC system, and formulate the corresponding optimization problems.

2.1 System Overview

Consider a UAV-assisted vehicle network, in which MEC servers are deployed on multiple macro base stations and UAVs to support delay-sensitive vehicle applications. Each MEC server stores a service cache to support a specific task. The road infrastructure is divided into different road segments, and each UAV runs at a consistent speed within its designated road segment without overlap with other UAVs. Each vehicle task can be further decomposed into multiple subtasks, and there are dependencies between each subtask. Vehicles can offload their computing tasks to macro base stations or UAVs within communication range. In addition, vehicles located in the overlap area between macro base stations and UAVs can only offload their tasks to one of the MEC servers.

2.2 Computation Model

Assuming that a vehicle-borne task is randomly generated by the vehicle, the computing task is represented by a DGA $G=(V,E)$. Each subtask of the vehicle-borne task consists of three components $\{q_{di}, q_{ci}, q_{ti}\}$, where q_{di} represents the amount of data that subtask v_i needs to transmit, q_{ci} represents the number of CPU cycles needed for finishing the subtask, and q_{ti} represents the maximum acceptable processing time for the subtask. $M=\{M_1, M_2,\cdots, M_S\}$ denotes the set of macro base stations, while $U=\{U_1, U_2,\cdots, U_S\}$ represents the set of UAVs .

To ensure that vehicles in the communication range of multiple macro base stations and multiple UAVs can only offload their tasks to one of the MEC servers, we let the binary variables b_{im_j} and b_{iu_j} be the vehicle-macro base station and vehicle-UAV association modes, respectively. If the vehicle unloads the subtask v_i to the macro base station m_j(UAV u_j), then $b_{im_j}=1$ ($b_{iu_j}=1$) otherwise $b_{im_j}=0$ ($b_{iu_j}=0$). Therefore, there is a constraint condition when the vehicle unloads the task to the macro base station:

$$\sum_{m_j\in M} b_{im_j}+\sum_{u_j\in U} b_{iu_j}=1, \forall i\in V \tag{1}$$

Now consider the computing resource model. Due to the higher transmitting power of the macro base station and the UAV compared to that of the vehicle and the relatively small data volume of processing results for each task, we will ignore the time consumption for descending processing results to each vehicle. We use R_c to represent the rate at which data are transferred between the vehicle and the MEC server. Then R_c can be expressed as:

$$R_c=s_c log_2\left(1+\frac{P_c\theta_c}{\sigma^2}\right) \tag{2}$$

The MEC server allocates bandwidth to the vehicle with s_c, while P_c represents the vehicle's transmitting power and θ_c represents the channel gain between the vehicle and the MEC server. The CPU main frequency of MEC server installed on the macro base station is denoted as f_m , and f_u represents the CPU main frequency of the MEC server on the UAV. The earliest start time of subtask v_i on the MEC server is defined as follows:

$$EST\left(v_i\right)=max\left\{\sum_{m_j\in M} T_{avail}\left(m_j\right)b_{im_j}+\sum_{u_j\in U} T_{avail}\left(u_j\right)b_{iu_j}, \max_{v_j\in Pred(v_i)}\left\{AFT\left(v_j\right)+\frac{q_{di}}{R_c}\right\}\right\} \tag{3}$$

where T_{avail} represents the earliest time when a MEC serve is in idle state. $AFT\left(v_j\right)$ is the actual completion time of v_j. $\frac{q_{di}}{R_c}$ represents the time required to transfer the

task data from the vehicle to the MEC server. For finishing the subtask v_i,The MEC server must have adequate computing resources and all predecessor tasks must be completed before the subtask v_i can be executed.Then the earliest completion time of subtask v_i can be defined as:

$$EFT(v_i) = EST(v_i) + \sum_{m_j \in M} \frac{q_{ci}}{f_m} b_{im_j} + \sum_{u_j \in U} \frac{q_{ci}}{f_u} b_{iu_j} \quad (4)$$

This creates a constraint condition for the execution of the subtask:

$$EFT(v_i) \leqslant q_{ti} \quad (5)$$

2.3 Energy Consumption Model

The system energy consumption can be divided into three parts: vehicle energy consumption, UAV energy consumption and macro base station energy consumption. Due to the limitation of battery capacity, the UAV energy consumption has an upper bound, denoted as U_{max}.Since we are concerned with the energy consumption in the process of the system's overall task processing, we ignore the energy consumption generated by vehicle movement. The same is true for the UAV energy consumption. Therefore, for task v_i, the energy consumption generated by the transmission is:

$$E_{tran}^{v_i} = \frac{q_{di} \cdot P_c}{R_c} \quad (6)$$

and the energy consumed by completing v_i is:

$$E_m^{v_i} = \sum_{m_j \in M} b_{im_j} f_m q_{ci} \xi_m + \sum_{u_j \in U} b_{iu_j} f_u q_{ci} \xi_u \quad (7)$$

where ξ is the effective capacitance parameter related to the CPU microarchitecture, and f is the main frequency of the CPU. The average energy consumption of running a CPU cycle is $\xi \cdot f$.

2.4 Problem Formulation

We present an optimization problem aimed at minimizing the delay and energy consumption, while still adhering to the previously established constraints. For the multi-agent problem, one solution is to set up a total central controller, each server sends its own information to the central controller, which will lead to extra time and energy consumption. Therefore, we propose a distributed solution to let each macro base station and UAV make decisions independently. Based on the model proposed in the last section, the optimization problem of the macro base station and UAV can be described as:

$$min \alpha AFT\left(v_{|V|}\right) + (1-\alpha) \sum_{i \in V} E_{v_i} \quad (8)$$

$$s.t. \begin{cases} (1)(7) \\ \sum_{i \in V} b_{iu_j} \left(f_u q_{ci} \xi_u + \frac{q_{di} \cdot P_c}{R_c} \right) \leqslant U_{max} u_j \in U \\ E_{v_i} = E_{tran}^{v_i} + E_m^{v_i} \end{cases} \quad (9)$$

3. MATD3-BASED RESOURCE MANAGEMENT SCHEME

Due to the NP-hard nature of the optimization problem discussed in the previous section, it cannot be solved in polynomial time. Traditional algorithms often struggle to find optimal solutions and lack generality, making it challenging to quickly solve these optimization problems using traditional methods. Therefore, We use reinforcement learning algorithms to deal with this problem.

3.1 Subtask priority

The first part of the algorithm aims to determine the priority of each subtask. Only by determining the priority of each subtask can we establish the execution order of each subtask, in order to make better decisions about task offloading later. Inspired by literature [13], in our scheme, we use a recursive approach to determine the priority of each subtask. Specifically, we first reverse all edges in the DAG $(v_i, v_j) \to (v_j, v_i)$. When $succ(v_i) = \varnothing$, we define $PV_{(v_i)} = 0$.Because we start from the sink subtask, and the priority of the sink subtask is set to 0. Then we recursively calculate the priority one by one. The specific formula is as follows:

$$PV_{(v_i)} = \max_{v_j \in succ(v_i)} \left\{ PV_{(v_j)} + \frac{1}{|U+M|} \left(\sum_{u_j \in U} \frac{q_{ci}}{f_{u_j}} + \sum_{m_j \in M} \frac{q_{ci}}{f_{m_j}} \right) + \frac{q_{di}}{R_c} \right\} \forall i \in V \quad (10)$$

3.2 Problem Transformation

In this section, we will convert the previously defined problem into a Markov game with partial observability, involving $U+M$ agents (comprising of M macro base stations and U UAVs). Next, we will explain the four elements of environmental state, observation, action and reward in detail.

1) *Environmental state*: The system environment state mainly includes vehicle position information and the position information of each MEC server, among which the altitude of the UAV's position should also be considered. The specific environment state is as follows:

$$
\begin{aligned}
s = \{ & x_{u_1}, x_{u_2}, \cdots, x_{u_U}, x_{m_1}, x_{m_2}, \cdots, x_{m_M}, x_{u_1}, x_{u_2}, \cdots, x_{u_U}, x_{m_1}, x_{m_2}, \cdots, x_{m_M}, \\
& x_{u_1}, x_{u_2}, \cdots, x_{u_U}, y_{u_1}, y_{u_2}, \cdots, y_{u_U}, y_{m_1}, y_{m_2}, \cdots, y_{m_M}, y_{u_1}, y_{u_2}, \cdots, y_{u_U}, \\
& y_{m_1} y_{m_2}, \cdots, y_{m_M}, y_{u_1}, y_{u_2}, \cdots, y_{u_U}, z_{u_1}, z_{u_2}, \cdots, z_{u_U}, x_1, x_2, \cdots, x_S \\
& y_1, y_2, \cdots, y_S, q_{d1}, \cdots, q_{ds}, q_{c1}, \cdots, q_{cs}, q_{t1}, \cdots, q_{ts} \}
\end{aligned} \tag{11}
$$

where x_i and y_i represent the x coordinate and y coordinate of vehicle i, x_{u_i}, y_{u_i} and z_{u_i} represent the x coordinate, y coordinate and z coordinate of UAV i respectively, and x_{m_i} and y_{m_i} represent the x coordinate and y coordinate of macro base station i.

2）*Observation:* we set that there is no information exchange between MEC servers, so the observed value can be described as:

$$
\begin{cases}
o_{m_i} = \{ x_{1,m_i}, x_{2,m_i}, \cdots, x_{j,m_i}, y_{1,m_i}, y_{2,m_i}, \cdots, y_{j,m_i}, x_{m_i}, y_{m_i}, q_{d1}, \cdots, q_{dj}, q_{c1}, \cdots, q_{cj}, q_{t1}, \cdots, q_{tj} \} \\
o_{u_i} = \{ x_{1,u_i}, x_{2,u_i}, \cdots, x_{j,u_i}, y_{1,u_i}, y_{2,u_i}, \cdots, y_{j,u_i}, x_{u_i}, y_{u_i}, z_{u_i}, q_{d1}, \cdots, q_{dj}, q_{c1}, \cdots, q_{cj}, q_{t1}, \cdots, q_{tj} \}
\end{cases} \tag{12}
$$

where $x_{j,m_i}(x_{j,u_i})$and $y_{j,m_i}(y_{j,u_i})$ respectively represent the x coordinates and y coordinates of vehicle j within the communication range of the corresponding macro base station m_i(UAV u_i).

3）*Action*: the action of each agent is obtained by the agent selecting an action from its action space according to its policy function and observation value. The action value can be described as:

$$
\begin{cases}
a_{m_i} = \left\{ b_{1m_i}, b_{2m_i}, \cdots, b_{sm_i} \right\} \forall i \in M \\
a_{u_i} = \left\{ b_{1u_i}, b_{2u_i}, \cdots, b_{su_i} \right\} \forall \in U
\end{cases} \tag{13}
$$

The binary variables b_{ju_i} and b_{jm_i} respectively represent whether vehicle j offloads the task to macro base station m_i or UAV u_i,while $bi, m(t) \in \{0,1\}$, $bi, j(t) \in \{0,1\}$.

4）*Reward*: As rewards guide each agent in selecting their optimal strategies which directly influences task offloading strategies for corresponding MEC servers, it underscores their critical importance throughout our algorithm. In the reward setting, we will give all agents the same reward every time.

3.3 MATD3-Based Solution

In order to tackle the multi-agent Markov game problem mentioned above, the MATD3 algorithm integrates the TD3 algorithm with a collaborative learning architecture for multiple agents. Next, we are going to introduce our proposed scheme based on MATD3.

1）*TD3 algorithm*:TD3 algorithm is a refined version of the DDPG algorithm. TD3 algorithm primarily addresses the issue of overestimating value estimates. When maximizing Q values with noise, the estimate function tends to continuously overestimate Q values. This can lead to inaccuracies in the estimate function as it learns to approximate the true value due to the presence of noise. Since these methods are based on the Behrman equation, updating the estimate function using subsequent states exacerbates this decline in accuracy. As a result, using an inaccurate estimate at each policy update can lead to accumulated errors and ultimately prevent convergence of the algorithm. Therefore, TD3 improves upon DDPG by incorporating ideas from Double DQN and implementing delayed learning and soft update methods for network parameter updates. Additionally, TD3 utilizes both exploration noise and policy noise during parameter updates for smoother policy expectations.

2）*System Framework:*Our MATD3 framework consists of a vehicle environment and agents, each utilizing the TD3 algorithm.During centralized offline training, agents have access to additional information observed by other agents, including their observations and actions. When updating parameters for both actor and critic, the actor selects an action based on its own policy function and local observation, which is then evaluated by the critic. In the training stage, since the information of all agents will be included in the environment state, the entire system environment is static for each agent. Since the participants are trained to select the best action from the action space only by their own local observation in the MATD3 algorithm, each agent can independently select the best action according to their own unique task offloading strategy and their own observation value in the execution stage. At the same time, since we set it in a multi-intelligence cooperation scenario, all MEC servers in the system work need to together to minimize the time of task completion and the energy consumed in the whole process as much as possible.

4. SIMULATION RESULTS AND ANALYSIS

In this section, we mainly compare the proposed scheme with the existing methods.We compares our method with

the DAGOA algorithm [16] and the TBTOA algorithm [12].

Figure 1 shows the delay and energy consumption under the three schemes with different number of subtasks. It is evident that the scheme we proposed has clear advantages in achieving lower average delay compared to the other two algorithms. Although the gap in energy consumption is small, the algorithm we proposed still performs the best. Fig. 1 respectively shows the average delay of the three algorithms under different service caching ratios and DAG task density. As shown in Fig. 1, with the increase of service caching ratio, the delay of all schemes decreases. This is because as the service cache ratio increases, all tasks have more offloading options.This reduces the number of cases in which a MEC server is idle but there is no corresponding service cache on the MEC server, causing the task to fail to offload. It can also be seen from Fig. 1 that, regardless of the service caching ratio, our proposed algorithm has an advantage in terms of low delay compared with the other two algorithms. Different from the service caching ratio, with the increase of DAG task density, the delay of the system will be higher. Because the increase of density represents that subtasks will have more dependencies, resulting in more subtasks being blocked in offloading. It can also be seen from Fig.1 that our scheme is better than the other two schemes regardless of the density.

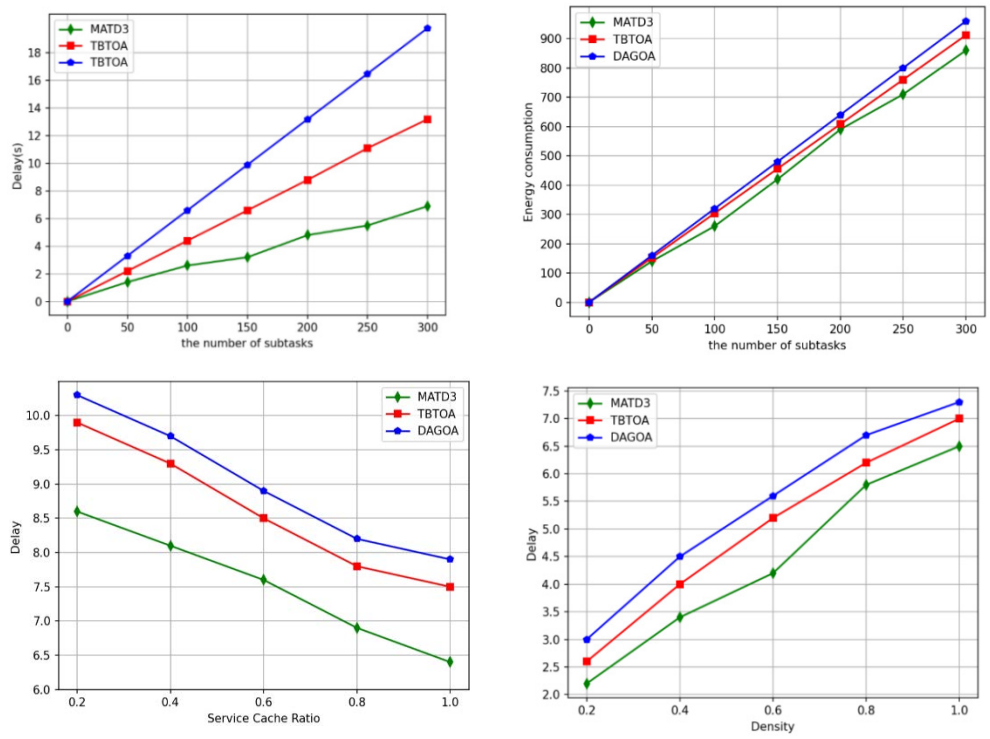

Figure.1 the delay and energy consumption of the three algorithms under different conditions

5. Conclusion

This study studies the task offloading strategy in the scenario of UAV-assisted vehicle network. We particularly focus on the task offloading problem in the context of service cache constraints.We formulate the corresponding optimization goal for this problem, which is to minimize the overall delay and energy consumption of the vehicle task by designing an appropriate task offloading strategy. This problem is NP-hard. Therefore, we design a solution by using MATD3 algorithm, and verify the effectiveness of our scheme through experiments. The experimental results show that our scheme is better than the existing schemes in reducing delay and energy consumption.

References

[1] K. Zhang et al., “Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks”, IEEE Access, vol. 4, pp. 5896-5907, 2016.

[2] Liu L, Zhao M, Yu M, et al. Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks[J]. IEEE Transactions on Intelligent Transportation Systems, 2022, 24(2): 2169-2182.

[3] Yang C, Liu Y, Chen X, et al. Efficient mobility-aware task offloading for vehicular edge computing networks[J]. IEEE Access, 2019, 7: 26652-26664.

[4] J. Huang, J. Wan, B. Lv, Q. Ye and Y. Chen, “Joint computation offloading and resource allocation for edge-cloud collaboration in Internet of Vehicles via deep reinforcement learning”, IEEE Syst. J., vol. 17, no. 2, pp. 2500-2511, Jun. 2023.

[5] S. Zhou, W. Jadoon and I. A. Khan, “Computing offloading strategy in mobile edge computing environment: A comparison between adopted frameworks challenges and future directions”, Electronics, vol. 12, no. 11, pp. 2452, 2023.

[6] C. Li, C. Qianqian and Y. Luo, “Low-latency edge cooperation caching based on base station cooperation in SDN based MEC”, Expert Syst. Appl., vol. 191, pp. 1-14, Apr. 2022.

[7] G. Li, M. Zeng, D. Mishra, L. Hao, Z. Ma and O. A. Dobre, “Latency minimization for IRS-aided NOMA MEC systems with WPT-enabled IoT devices”, IEEE Internet Things J., vol. 10, no. 14, pp. 12156-12168, Jul. 2023.

[8] Y. Han, Z. Zhao, J. Mo, C. Shu, and G. Min, “Efficient task offloading with dependency guarantees in ultra-dense edge networks,” in 2019 IEEE Global Commu-nications Conference (GLOBECOM), 2019, pp. 1–6.

[9] X. Fu, B. Tang, F. Guo, and L. Kang, “Priority and dependency-based dag tasks offloading in fog/edge collaborative environment, ” in 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2021, pp. 440– 445

[10] S. Sundar and B. Liang, “Offloading dependent tasks with communication delay and deadline constraint”, Proc. IEEE Conf. Comput. Commun., pp. 37-45, 2018.

[11] G. Zhao, H. Xu, Y. Zhao, C. Qiao and L. Huang, “Offloading dependent tasks in mobile edge computing with service caching”, Proc. IEEE Conf. Comput. Commun., pp. 1997-2006, 2020.

[12] Lv X, Du H, Ye Q. TBTOA: A DAG-Based Task Offloading Scheme for Mobile Edge Computing[C]//ICC 2022-IEEE International Conference on Communications. IEEE, 2022: 4607-4612.

[13] Y. Sahni, J. Cao, L. Yang and Y. Ji, “Multihop offloading of multiple DAG tasks in collaborative edge computing”, IEEE Internet Things J., vol. 8, no. 6, pp. 4893-4905, Mar. 2021.

[14] Wang Z, Sun G, Su H, et al. Low-latency scheduling approach for dependent tasks in MEC-enabled 5G vehicular networks[J]. IEEE Internet of Things Journal, 2023.

[15] Peng H, Shen X. Multi-agent reinforcement learning based resource management in MEC-and UAV-assisted vehicular networks[J]. IEEE Journal on Selected Areas in Communications, 2020, 39(1): 131-141.

[16] Han Y, Zhao Z, Mo J, et al. Efficient task offloading with dependency guarantees in ultra-dense edge networks[C]//2019 IEEE Global Communications Conference (GLOBECOM). IEEE, 2019: 1-6.